Radboud University Nijmegen

Faculty of Social Science

FlowCoder

A Model of Symbolic Thought Composition via Program Synthesis using Generative Flow

Thesis MSc Artificial Intelligence

Student Information

Surname:	Hommelsheim
First name:	Ron
Student number:	s1000522
E-mail address:	ron.hommelsheim@ru.nl
Course code:	SOW-MKI94 (45 EC)
AI Specialisation:	Cognitive Computing

Supervisor Information

Role:	Supervisor
Surname:	Thill
First name:	Serge
Institute:	Radboud University
E-mail address:	serge.thill@donders.ru.nl
Supervision type:	Internal

Notes

Acknowledgement

Abstract

I posit that the construction of our conceptual architecture, abides by the same principles as any hierarchical self-organising system, namely discernment about what it is and what it is not.

Contents

List of Figures

List of Tables

1 Introduction

One of the most profound questions one can ask, is that of his own cognition. How is it possible that we have a representation of the world such that we can dream, imagine, make quick inferences, come up with hypotheses, understand causal relations, have multiple representations of concepts and intrinsically understand relations between concepts?

In this thesis, I address these capacities of human cognition and outline how we can interpret them in a Bayesian framework.

In this thesis I want to investigate and discuss a certain model of cognition. - We look at the world and learn to distinguish things. Discernment is the fundamental operator of cognition. we create a model of the world and of ourself (res cogitans, res extensa). - this model is like a game engine in the head, (simulation simulacrum). we have assets, etc. and using these concepts we can construct ideas, dreams, memories, reality, etc. its all made from the same stuff.

- I think that we receive sensory information and compute statistical regularities over those sensory inputs.
- we construct a model of internal information and of external information and separate the world from the self, with ever more fine detail. - we create a generative model (reflexive) and inference model (reasoning)
- we are able to extract concepts into variables (symbols) and compute using these symbols.
- i think that the conceptual framework has a similar architecture as any self-organizing system (holarchic)
- dehaene et al show some ideas of probabilistic programming etc.
- First I discuss necessary (but not sufficient) capabilities of human cognition.
- Then I present a model of how to build it (PPL)
- And then I propose suggestions of how to improve it and build a model

1.1 Main Contributions

The main contributions of this thesis are:

- I argue that the same multi-scale hierarchies that govern biological organization can be applied to the conceptual realm
- I develop a method of bayesian program synthesis, using GFlowNet
- I present the philosophical ramifications
- Have a running example throughout the text.
- check all [sources] etc.
- question: should i put prompts to gpt for all capabilities in the intro? (abduction, analogies, etc.)
- pseudocode from other paper?

- number all equations?
- check that forward policy has ϕ everywehere and generative model θ
- also check notation $\pi(s_t + 1...)$ or $\pi(z, ...)$
- change replay_prob to greek letter?
- Check what whould be in intro vs methods.
- Introduction should introduce all capabilities and also the background work. and especially tell why we introduce it.
- Should parameterisation section (e.g. transformer) be more detailed?
- LLMs should be shortened and put in back discussion (only keep relevant parts in intro)

1.2 Cognition

In the following I will introduce some necessary (but not sufficient) core concepts of human cognition and show how we can gradually build up a model that explains some of these capabilities. Then I will show how I can improve it and explain my own model FlowCoder.
poverty of the stimulus. we learn from only a few observations.

1.2.1 World model

Imagine your brain as an interactive game engine. Just as a game engine generates dynamic virtual environments, complete with rules and physics that players interact with, the brain constructs a model of the real world. This model includes rules (physical laws, social norms), entities (objects, people), and interactions (how things work and relate to each other). We learn to navigate and predict our environment, constantly updating our internal model based on new experiences and information. This analogy, introduced by Tenenbaum et al. [4, 5, 6] extends beyond mere perception, encompassing imagination, dreams, and memory. Each of these cognitive functions can be seen as manifestations of the brain's ability to generate, manipulate, and explore various scenarios and possibilities within its internal model. Dreams and imaginative constructs, while seemingly detached from reality, are composed of the same 'material' as our waking perceptions – they are all products of the brain's intricate simulation capabilities. [sources] The self, in this view, becomes both a creator and a perceiver of its subjective reality, a reality that, while grounded in the external world, is ultimately shaped by the mind's interpretative and predictive faculties.

This paradigm posits that the brain is not merely a passive recipient of sensory inputs but rather an active participant in the construction of reality. Through the lens of Bayesian inference, the brain is conceptualized as a predictive machine. It constantly generates hypotheses or predictions about the world, drawing on past experiences and current sensory data. The brain, rather than simply processing incoming sensory information, actively infers probable causal factors behind this data, effectively reverse-engineering the world. This inferential process is guided by Bayesian principles, where the brain weighs the likelihood of various hypotheses against the evidence presented by sensory inputs. The 'prediction error' – the discrepancy between these predictions and actual sensory experiences – becomes a crucial component, informing and refining the brain's model of the world. This is the premise of many Bayesian cognitive models such as the Free-Energy.

Principle, (Hierarchical) Predictive Coding, Bayesian Brain Theory, and many other variants [7, 8] [include more sources]. The concept of the 'game engine in the head' aptly encapsulates this process, suggesting that our cognitive system operates similarly to a simulation engine, continuously constructing and updating our mental representation of the world.

Talk about intuitive physics and intuitive psychology or other research of human capabilities?

- Intuitive physics etc.

1.2.2 Bayesian Model

The fundamental tenet of Bayesian computational cognition is that the brain interprets the world by forming probabilistic models and updates them according to Bayesian inference principles. This means the brain weighs prior beliefs (previous experiences and knowledge) and the likelihood of new sensory evidence to arrive at posterior beliefs (updated model of the world). The brain uses these posterior beliefs to make predictions about future events, thus enabling adaptive behavior. Bayesian inference can be formalized by:

$$P(z|x) = \frac{P(x|z) \cdot P(z)}{P(x)} \tag{1}$$

Where:

- $P(z|x)$ is the posterior probability of hypothesis or latent variable z given the observed data x.
- $P(x|z)$ is the likelihood of data x given that hypothesis z is true.
- $P(z)$ is the prior probability of hypothesis z.
- $P(x)$ is the marginal likelihood, the probability of the data x.

1.2.3 Compositionality

Humans possess a native capacity for meta-cognitive evaluation, instinctively gauging the reliability of their own knowledge [5]. Such self-assessment serves as a compass for learning, steering the acquisition of new information in a manner that refines and optimizes our cognitive architectures. Compositionality is central to cognitive productivity and learning. It allows for the creation of infinitely varied representations from a finite set of basic elements, similar to the mind's capacity to think an endless array of thoughts or generate countless sentences. This is particularly useful in forming hierarchical structures that simplify complex relationships, thus making inductive reasoning more efficient. In Bayesian terms, this means that the latent z can be compositional and take the form of a nested hierarchical structure.

should holarchies go here?

out of distribution generalisation (OOD)

1.2.4 Causal Models

Humans are endowed with an intuitive grasp of causality, which functions as the underpinning for both predictive and counterfactual reasoning [5]. This enables humans to extrapolate beyond the confines of empirical data, to conjure hypothetical worlds e.g. in imagination, play, and dreaming. A causal model typically consists of a set of variables and a set of directed edges between these

variables, often represented as a Directed Acyclic Graph (DAG). The vertices in this graph denote variables, which, in this context, could be seen as cognitive states. The directed edges signify causal relationships, pointing from cause to effect. Causal models facilitate interventions, counterfactuals, and causal inference. Intervention is the ability to model the outcome of purposeful changes, represented mathematically as "do" operations [9]. For example, if one were to intervene to set $X = x$, the model would enable us to compute the distribution over Y post-intervention. Counterfactual reasoning allows us to traverse back in time, in a sense, and assess alternative realities — what would have happened to Y if X had been different? Pearl sees this faculty as a crucial component of human cognition [9].

1.2.5 Abduction

How do we come up with the best explanation given partial information and limited resources? This is the problem of abduction, which can be seen as a collective term to both the generation of hypotheses, in which case it is termed *abduction proper*, as well as the justification of hypotheses, also termed *Inference to the Best Explanation (IBE)* [10]. The main problem associated with abduction proper is the difficulty of assessing how one generates fitting hypotheses quickly, while ignoring all possible non-sensical hypotheses. Inference to the Best Explanation entails: given that we already have a set of candidate hypotheses, how do we pick the best one? Furthermore, how do we define "best"? Kwisthout presents a computational model in which he characterises "best" as an explanation that maximizes both probability as well as information content while being parsimonious [11].

In the Bayesian framework the marginalisation term $P(x)$ represents the probability of the observed data, and calculating it requires integrating over all possible hypotheses. In practical terms, this means considering how each potential hypothesis might account for every piece of observed data. This is intractable. To manage this challenge, Bayesian approximation methods provide ways to approximate $P(x)$ without needing to perform exhaustive calculations. One common approach is to use sampling methods, such as Monte Carlo methods, which estimate $P(x)$ by taking random samples from the probability distributions of the hypotheses. Another approach is variational inference, which approximates the probability distributions with simpler ones that are easier to compute. [sources + for a detailed discussion see section x].

It is common to divide between the generative model and the recognition density. In this Bayesian framework, the joint probability $P(z, x) = P(x|z) \cdot P(z)$ is referred to as the generative model that hypothesizes how sensory data are generated by the hidden states of the world [12]. In other words, it specifies a joint probability distribution over sensory inputs and potential causes of those inputs. These causes can be anything from the presence of objects in the environment to more abstract concepts like social cues. The recognition model is an approximation of the posterior $Q(z|x) \approx P(z|x)$, the brain's inference about the state of the world given the data. [for further discussion see]

1.2.6 System 1 & System 2

Kahneman describes two distinct systems that characterize the dual aspects of human cognition [13]:

System 1 operates automatically and quickly, with little or no effort and no sense of voluntary control. It encompasses intuitive judgments and perceptual associations — it allows for rapid,

heuristic-based processing that is often subconscious. System 1 is akin to the generative model in Bayesian inference. Just as System 1 can produce instantaneous responses and intuitions, the generative model provides immediate perceptual hypotheses and predictions that guide behavior in a fluid and dynamic manner without the need for conscious deliberation.

System 2 allocates attention to the effortful mental activities that demand it, including complex computations. It is associated with the conscious, rational mind and is deliberate, effortful, and orderly. The process of updating the recognition density is more reflective and can be related to the slow, controlled inference that characterizes conscious thought. Adjusting the recognition density is akin to the effortful correction or modulation of System 1's rapid predictions when errors are detected or when more complex reasoning is required.

For example, when learning to play the piano, initial efforts require conscious attention and deliberate practice to master certain phrases (System 2). Over time, as these phrases become internalized, they can be executed more automatically and intuitively (System 1). This transition illustrates how learned skills move from conscious, effortful control to intuitive, automatic execution, mirroring the interplay between the recognition density (System 2) and the generative model (System 1) in Bayesian cognition.

1.2.7 Varied Representations

[aristotle - it is a sign of intelligence to be able to entertain a thought without accepting it.] Humans interpret the same data in different ways. Optical illusions, like the old woman/young woman or the duck/rabbit images [show link], serve as a metaphor for this phenomenon. They demonstrate how one set of data (the image) can lead to multiple interpretations. In AI, this reflects the system's capacity to process and understand information from multiple perspectives.

From a Bayesian perspective, this idea relates to the choice between seeking a singular, most probable interpretation (Maximum A Posteriori, or MAP: $\arg max P(z|x)$) and considering a range of possible interpretations (approximating the posterior distribution: $P(z|x)$). The former approach aims for a definitive answer, while the latter embraces uncertainty and variability in the data.

1.2.8 Concepts

Concepts, as the building blocks of thought, vary widely in their forms and representations. They range from concrete, perceivable objects such as 'a table' to abstract notions like 'love' or 'money', theoretical constructs ('the gene', 'the atom'), mathematical entities ('addition', 'multiplication'), and even non-existent entities like 'a unicorn'. This variety, as highlighted by Blouw et al. [3], underscores the diverse spectrum of concepts that the human mind can conceive and manipulate.

An effective conceptual framework not only categorizes these varied forms but also explains their *extension* (why a concept refers to certain things and not others) and *intension* (why it describes these referents in specific ways) [14].

Several key criteria help in mapping and understanding this concept space:

Relationship Path This refers to the connections between concepts, governed by their similarities, differences, and functional relationships. It explains how one concept is related to or can be derived from another.

Mutual Information This criterion examines how well concepts predict each other, based on their co-occurrence and interaction in reality. It reflects the degree to which the presence or understanding of one concept informs about another.

Change Distance This involves measuring the conceptual 'distance' or difference between two concepts. It accounts for the extent of transformation or cognitive shift required to move from one concept to another in the conceptual space.

Parameter Distance This refers to the number of 'knobs' or minimal changes needed to transition from one concept to another. For example, determining the minimal edits or shifts needed to transform the concept of 'dog' to 'cat', or 'rock' music to 'jazz'. This concept is akin to edit distance in text or Kolmogorov complexity in information theory, offering a quantifiable measure of conceptual transformation.

need sources for all of these

this section needs to be rewritten. its shit

[kiki, bobo, other examples]

[for further discussion see]

1.2.9 Large Language Models

This should be Concept Space not LLMs

In his recent book "What is ChatGPT doing?", Stephen Wolfram analyses the internal mechanisms of Large Language Models (LLMs) such as GPT [15]. LLMs are descendents of the Transformer architecture, originally introduced in 2017 by Vaswani et al. [16] and are trained on massive datasets of text and code. This allows them to learn the statistical relationships between words and phrases, and to generate human-quality text, translate languages, write different kinds of creative content, and answer questions in an informative way. Embedding is the process of converting tokens into dense vectors of numbers. These vectors represent the semantic and syntactic information about the tokens. The embedding layer is important for LLMs because it allows them to learn the relationships between tokens in a high-dimensional space. Various levels of embeddings are encoded, which gives aggregate words, sentences, paragraphs, etc. positions in this latent space. LLMs primarily learn and operate through next token prediction. They use statistical patterns in data to predict language use, context, and meaning, focusing on determining the next word or phrase in a sequence. This predictive training over time leads to a complex internal representation of language, enabling the model to produce responses that are grammatically correct, contextually relevant, and semantically nuanced. Wolfram highlights that GPT doesn't have any explicit knowledge of grammar and the parse trees that govern natural language, yet they learn these nested syntax trees implicitly. Nonsense sentences such as Chomsky's famous example "Colorless green ideas sleep furiously" [source], or jabberwocky sentences [source], show that there is more to language than syntax. There seems to be a semantic grammar, which LLM models learn implicitly. LLMs, through exposure to vast amounts of text containing conditional and logical structures (e.g., "if X, then Y"), develop an ability to recognize and generate logically coherent sentences. They seem to mimic forms of syllogistic reasoning, as seen in sentences like "All X are Y. This is not Y, so it's not X." (as in "All giraffes are tall. This is not tall, so it's not a giraffe."). It's possible that LLMs have "discovered" syllogistic logic by looking at huge amounts of text. However, when it comes to more sophisticated formal logic, they fail [source], since they learned this implicit logic

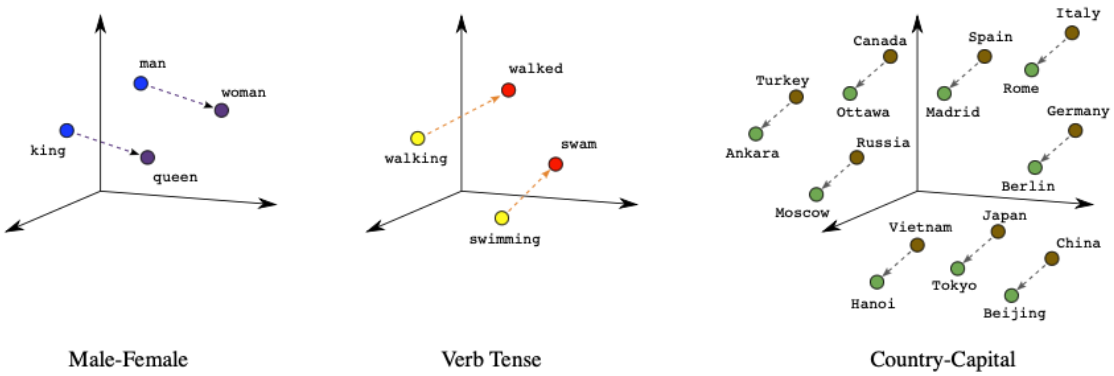

Figure 1: taken from google

by correlation and they don't "know" the actual rules of logic. The concept space that LLMs navigate can be illustrated through analogies such as "man is to king as woman is to queen." This represents the learnt semantic space, where relationships and associations are encoded in a high-dimensional embedding space. In this space, concepts and words are positioned in such a way that their relative distances and directions encode meaningful relationships, mirroring the complexities of human language and thought.

Explain the figure, explain vector space, dot product, etc.

[for further discussion see]

- holarchic direction sensory data to features, etc.

until here i show how we build up the elements needed for this framework of bayesian program synthesis., which combines almost all these things.

1.2.10 Probabilistic Programming as a language of thought

Dehaene et al. posit that human cognition is uniquely characterized by its ability to form symbolic representations and recursive mental structures akin to a language of thought, enabling the creation of domain-specific conceptual systems [1]. This cognitive ability allows for the generation of new concepts through the compositional arrangement of existing elements, a process exemplified by the derivation of geometric concepts. Cognition simplifies complex patterns into mental representations via mental compression, where the complexity of a concept is measured by the length of its mental representation as per the Minimum Description Length (MDL) principle.

Neuroscientific research indicates the presence of specialized brain circuits responsible for processing different domains of these languages of thought, with certain brain regions involved in linguistic processing and others in non-linguistic domains like mathematics and spatial reasoning [source]. The recognition of mathematical patterns is linked to the ability to detect repetition with variation, a process underpinned by specific neural areas that vary with cognitive domain.

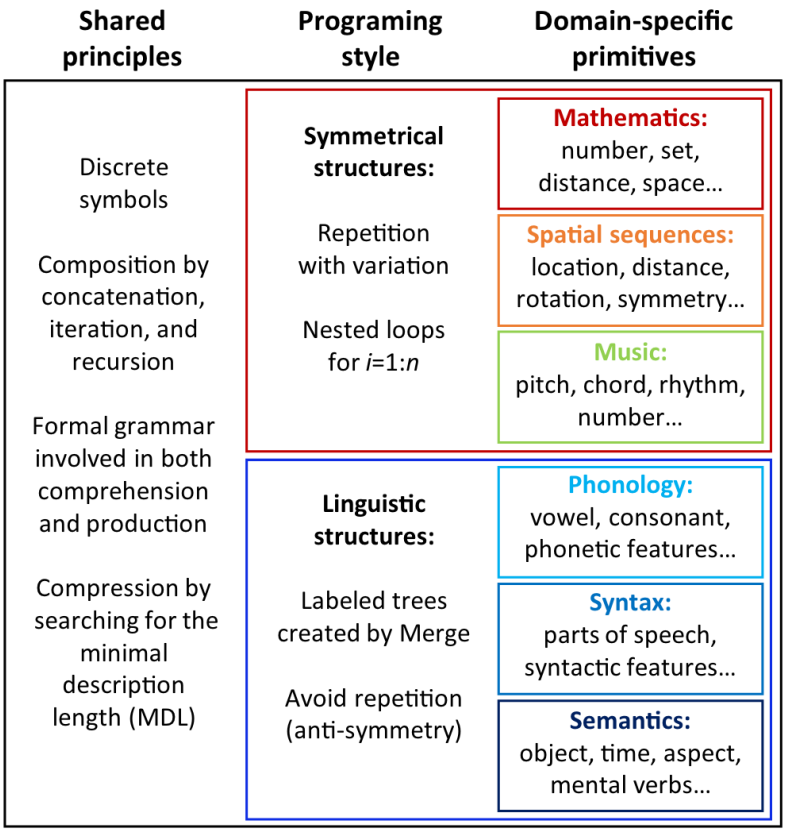

Figure 2: Image from [1].

Explain figure

Distinct from other primates, humans have developed the capability to use symbols in complex, rule-based systems, highlighting a unique aspect of human cognitive development. Non-human primates may associate signs with concepts; however, they do not appear to use these in the recursive, rule-based manner that humans do.

[Tenenbaum et al] posit that the brain implements mechanisms analogous to those found in probabilistic programming languages, enabling it to represent and infer the probabilistic structure of the world. Probabilistic programming provides a framework for defining complex probabilistic models and for performing inference in these models, and the hypothesis is that the brain engages in similar computational processes. [multiple sources]

Experiments show that humans do not seem to start from blank-slate but rather from rich domain knowledge [argument for primitives, more sources] [17]. Lake et al. propose that concepts can be represented as simple stochastic programs [elaborate]. A program here can be thought of a procedure that generates more examples of the same concept. If a program would represent the concept "animal", if would generate examples such as "giraffe", "zebra", "fish", and so on. Of course, higher-level programs could produce lower-level programs, in other words, in this paradigm, the essential aspect of compositionality gives rise to a part-whole hierarchical structure, i.e. a holarchy [reference].

child as a hacker

[the assumption we make here is that features are somehow aggregated or extracted into symbols [see [18]], primitives, along with possibly innate symbols [5]]

1.3 Background

1.3.1 Program Synthesis

Program synthesis is undecidable [19]. This means that some form of search over the program space is necessary. A variety of techniques have been investigated to overcome this challenge, ranging from evolutionary algorithms, to MCMC sampling, probabilistic inference, and most recently neurosymbolic architectures, making use of neural network to help guide the search through a vast program space [19, 20].

Robustfill [21]

In the following, I will first introduce the state-of-the-art model in this regard, to present the possibilities of this framework as well as the methods used, many of which inspired my own model.

- neural guided search, neuro-symbolic architectures, etc.

1.3.2 DreamCoder

One of the most successful models in this endeavor is DreamCoder - a model that synthesises programs from initial primitives and a set of tasks with the objective of learning its own domain specific language [2]. The model utilises a modified version of a wake-sleep algorithm introduced by Hinton to learn a generative model and a recognition network in tandem, following the paradigm of separating the world model from the inference model [22] [reference to earlier section, sys 1,2]. The generative model learns a probability distribution over programs while the recognition network learns to map from tasks to programs in order to perform neurally guided search over the program space. Using this recognition model, they build a parallel search algorithm which is a mixture of best-first and depth-first and enumerates programs with decreasing probabilities. See [23] for details. Commonly used sub-routines, are chunked and abstracted into concepts which become more accessible. This narrows the search tree immensely and enables scalability. In sum, abstraction narrows the depth, while the recognition model narrows the breadth of the search space.

Tasks can be generative (e.g. creating an image) or conditional, (input-output relation, e.g. sorting a list). Figure 3(A) shows examples of tasks across different domains. Figure 3(B) shows an example, in which the task to learn is to sort a list. On the left we see the initial primitives. The shaded region shows the learned library of concepts. On the right we see the final solution, which uses `concept15`, which itself uses previously abstracted concepts. Note the difference in the length of the program using abstracted concepts vs. only initial primitives, shown beneath.

The recognition model Q takes a task as input and outputs a 3-dimensional bigram transition matrix Q_{ijk}, where i indexes possible children, j indexes possible parents and k indexes which argument of the parent is being generated. This parameterization was chosen as a balance between quality and efficiency.

should this be here? or is it too detailed?

1.3.3 DeepSynth

In their 2021 paper, Fijalkow et al. proposed a framework called "distribution-based search", in which they investigate the difficult problem of searching through a DSL to find programs matching a specification in a vast hypothesis space [24].

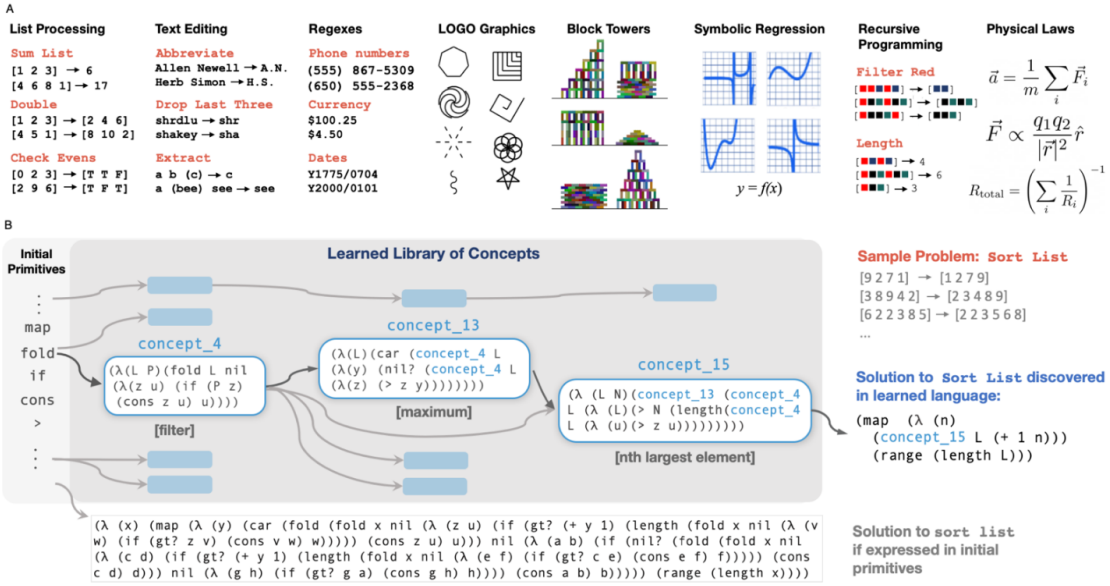

Figure 3: (A) 8 different domains of tasks. (B) Representation of the learned library of concepts. On the left we see the initial primitives from which the concepts in the middle region are composed. On the right we see a task as input-output relations and the found solution. On the bottom is the same solution expressed only with initial primitives. Image taken with permission from the original paper [2].

They introduce DeepSynth [1], a general-purpose program synthesizer which constructs programs from input-output examples, and a useful framework allowing us to test different models and search methods (see section x for details).

The authors discuss different program finding strategies. Specifically, they find that both enumerative search (as in DreamCoder) and sampling are viable strategies, where search is associated with prioritising quantity, i.e. creating many programs quickly, whereas sampling strategies prioritise quality but may be slower, since resampling may occur. An additional benefit of sampling over search is space efficiency - already created programs don't need to be memorised.

1.3.4 GFlowNet

Generative Flow Networks (GFlowNets), introduced by Bengio et al., are a class of generative models designed to learn to construct objects from a target distribution over complex high-dimensional spaces, particularly where explicit density estimation is challenging and diverse candidates are encouraged [25]. GFlowNets learn a stochastic policy for generating sequences of actions that lead to the construction of a sample. This sampling process is done in such a way that the frequency of generating any given sample is proportional to a given reward function associated with that sample. A trained GFlowNet can therefore act as an excellent sampler.

GFlowNet is still quite a new model, first published in 2021. The literature on new models including improvements and adjustments is growing rapidly.

[different way of dealing with marginalization, using flow.] what are the added benefits? Include GNN-GFNs, using them for building molecules. Any problem which starts wich building blocks that can be composed to more complex structures. See Dehaene, as well as the discussion on

[1] https://github.com/nathanael-fijalkow/DeepSynth

holarchies.

1.3.5 Program Embedding

In section 1.2.9 we have seen that the underlying mechanism of the Transformer architecture used in LLMs, is able to learn syntactic and semantic structure from its data. Kim et al. discuss the benefit of a distributed representation of PCFGs, over the classic notion of simply assigning a scalar to each rule of the CFG to attain a PCFG [26]. To that extent I will utilise the Transformer as the generative model.

1.4 Aim

We have seen how a probabilistic program synthesis algorithm looks like and what it can achieve [change this sentence]. Moreover, we have learnt that sampling might be a good alternative to enumerative search and that GFlowNets are a method to implement a generative model and sampler.

Additionally, we have learnt that LLMs which use the Transformer architecture [embeddings plus self-attention instead of transformer architecture?] learn statistical regularities that go beyond syntax into the semantic realm. In the following I will show how I combine all these concepts to arrive at my own model FlowCoder [2].

Why do we do it?

1. First, I want to investigate whether GFlowNet is an adequate strategy within the realm of program synthesis. Can a GFlowNet be trained to be a good sampler?
2. out of distribution generalisation
3. Another reason for using GFNs over DreamCoder or other methods is that we build programs sequentially. This makes the method more dynamic and gives us more control. For example we can do inferences about intermediate latent variables, train from partial states, etc. [should this be here or in discussion?]
4. We want to embed programs. This gives us an additional semantic layer. [compare with neural pcfg, and other approaches and criticisms of dreamcoder]
5. I want to learn the whole probabilitiy dist to a task, not just the MAP.
6. DreamCoder has most o the cognitive capabilities, but not program space (arguably), MAP, different way of dealing with marginalisation, although it is amortized variational inference, and sampling rather than enumeration (no syntactic PCFG, rather embedded neural PCFG.)

Research Questions: what do we want to find out

1.

[2] Code available at `https://github.com/R1704/master_thesis`

2 Methods

I am following the assumption that we don't start from a blank slate but from some initial concepts, here operationalized as primitives, fundamental program components. These initial primitives comprise a minimal domain-specific language (DSL) and can be composed into more complex programs, which represent the causal structure generating the observations we perceive. As such, an agent learns its own DSL, which in terms of a language of thought is analogous to inferring a mental grammar and using it to learn new concepts.

2.1 Computational Model

2.1.1 DeepSynth Framework

Both in DreamCoder and DeepSynth, programs are represented as abstract syntax trees (ASTs). An AST is a tree representation of the syntactic structure of the program, with nodes representing operations or primitives and edges representing their compositional relationships. In order to avoid predicting variables, deBruijn indexing is used, which is a technique to represent bound variables in a way that avoids naming conflicts and simplifies variable substitution [27]. Named variables are replaced with numeric indices representing the number of enclosing λ abstractions that bind the variable. See figure 6 for a visualization of an AST using deBruijn indexing.

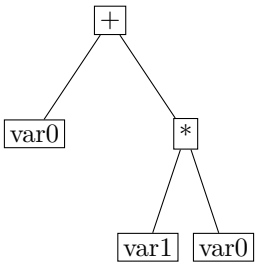

Figure 4: An example of an abstract syntax tree (AST) using deBruijn indexing. This translates to the function `f = var0 + var1 * var0`.

Here, an initial DSL along with suitable syntactic constraints compile into a context-free grammar (CFG), which defines the possible structures of programs within its DSL. A CFG consists of a set of production rules that describe how to generate strings from a set of non-terminal and terminal symbols. It's "context-free" because the production rules are applied regardless of the surrounding symbols. In DeepSynth, a prediction model is used to predict weights for a probabilistic CFG (PCFG), extending the CFG by associating probabilities with the production rules. This allows the grammar to not only generate the syntactic structure of a program but also to represent beliefs about the relative plausibility or frequency of different structures [3]. This is similar to DreamCoder's transition matrix which is outputted by the recognition model. The PCFG guides the search and inference process towards more likely programs. DreamCoder however, does not specifically use a PCFG. Both frameworks employ a typed λ-calculus, hence there are restrictions on program arguments, etc. (syntactical constraints). DreamCoder performs type inference during program generation. To spare computational cost, DeepSynth constructs the CFG beforehand which in turn increases the size of the CFG.

[3] See appendix 4 for a formalization of CFGs and PCFGs.

2.1.2 GFlowNet

In the following I will give a detailed explanation of GFlowNet

should this be here or in intro? Explain why we explain this. 1. to show how the algorithm works, 2. to show how we deal with the marginalization term

GFlowNets create a directed acyclic graph (DAG) over the state space, where vertices correspond to states or partial samples and edges denote transitions or adding a component to a partial sample, in which the edges carry a flow from source to targets [28, 25].

show diagram of DAG and talk about the causality requirement! (In the methods section we explain what we do and why we do it.)

In GFlowNets, the "flow" in the network corresponds to the process by which the network constructs a sample, which can be thought of as a path in a graph where nodes are partial samples, and edges correspond to adding a component to the partial sample. The core training objective for a GFlowNet is to satisfy the flow matching constraint. The idea is to ensure that the flow into any state (a partially constructed sample) should match the flow out of it, given the reward associated with complete samples. The flow here refers to the expected transitions into or out of a state under the model's stochastic policy. Formally, a state s represents a partial object a certain stage in the generative process. A trajectory τ is a sequence of states $s_0, s_1, ..., s_T$ that the model traverses from an initial state s_0 to a terminal state s_T, where the target structure is complete. A trajectory τ is formed by a sequence of actions $a_1, a_2, ..., a_T$, where each action a_t transitions the model from state s_{t-1} to state s_t. The sequence of actions is governed by a policy π, which defines the probability of choosing a particular action given the current state. The flow $F(\tau)$ of a trajectory τ is defined as the product of the probabilities of each transition along the trajectory, multiplied by the reward $R(s_T)$ of the terminal state, normalized by a partition function Z.

$$F(\tau) = \frac{R(s_T)}{Z} \prod_{t=0}^{T-1} \pi_\theta(s_{t+1}|s_t) \tag{2}$$

The partition function Z ensures that the sum of flows over all possible trajectories equals one, effectively normalizing the distribution. Since we don't know Z, we can estimate it by parameterizing it as Z_θ. The flow matching constraint enforces that for any given non-terminal state s, the total flow into s must equal the total flow out of s:

$$F(\tau) = F(\tau') \tag{3}$$

where $F(\tau')$ is the reverse trajectory. We can utilize this property to create a suitable loss function to train the GFlowNet [29]. Combining equations 2 and 3 gives us:

$$\frac{R(s_T)}{Z_\theta} \prod_{t=0}^{T-1} \pi_\theta(s_{t+1}|s_t) = \frac{R(s_0)}{Z_\theta} \prod_{t=0}^{T-1} \beta_\theta(s_t|s_{t+1}) \tag{4}$$

Here β is the backward policy, predicting parent states. The initial state s_0 has the total flow and no reward, we can rewrite it and get:

$$Z_\theta \prod_{t=0}^{T-1} \pi_\theta(s_{t+1}|s_t) = R(s_T) \prod_{t=0}^{T-1} \beta_\theta(s_t|s_{t+1}) \tag{5}$$

We can now take the log on both sides:

$$\log\left(Z_\theta \prod_{t=0}^{T-1} \pi_\theta(s_{t+1}|s_t)\right) = \log\left(R(s_T) \prod_{t=0}^{T-1} \beta_\theta(s_t|s_{t+1})\right) \tag{6}$$

This can be rearranged to:

$$\log Z_\theta + \sum_{t=0}^{T-1} \log \pi_\theta(s_{t+1}|s_t) = \log R(s_T) + \sum_{t=0}^{T-1} \log \beta_\theta(s_t|s_{t+1}) \tag{7}$$

The trajectory balance loss is the squared difference:

$$\mathcal{L}_{TB} = \left(\log Z_\theta + \sum_{i=0}^{T-1} \log \pi_\theta(s_{t+1}|s_t) - \log R(s_T) - \sum_{t=0}^{T-1} \log \beta_\theta(s_t|s_{t+1})\right)^2 \tag{8}$$

In order to mitigate the computational [cost], I am embedding all rules rather than primitives this is explained in a later section , such that each rule is unique. Thus, every predicted node (rule) has exactly one parent, in other words, I am essentially linearising the tree. Therefore, β will always be 1 and can be disregarded from the equation. Moreover, since we are solving tasks $x \in X$, we have a conditional reward distribution $R(s_T|x)$, as well as a conditional forward policy $\pi_\theta(s_T|x)$ and partition function $Z_\theta(x)$. This gives us the final loss:

$$\mathcal{L}_{TB} = \left(\log Z_\theta(x) + \sum_{t=0}^{T} \log \pi_\theta(s_{t+1}|s_t, x) - \log R(s_T|x)\right)^2 \tag{9}$$

We need to formalize the overall objective

2.1.3 Expectation-Maximization

Hu et al. introduce a GFlowNet-EM which utilizes Expectation-Maximization in order to deal with the difficult problem of joint optimization [30] [4]. The core principle of Expectation-Maximization (EM) involves an iterative two-step process: the Expectation (E) step computes an expectation of the log-likelihood evaluated using the current estimate for the parameters, and the Maximization (M) step, computes parameters maximizing the expected log-likelihood found in the E-step. The GFlowNet version is slightly different and we separate the parameterization. In the E-step the recognition model is optimized, i.e. the forward policy of the GFlowNet that is proportional to the posterior. Here, empirical data as well as a trajectory is sampled and the trajectory balance loss is used to update the parameters of the forward policy $\pi_\phi(z|x)$, i.e. $\nabla_\phi \mathcal{L}_{TB}$. In the M-step also empirical data and a trajectory is sampled but only the reward is used to update the parameters of the generative model $p_\theta(x|z)p_\theta(z)$.

2.1.4 Sleep

As sleep has been proven to be a crucial element of the models success, I am utilizing it too.

Replay In Replay, I am training the forward policy on previously correctly solved task-program pairs (x, z), using the trajectory to guide the model to the correct solution and optimizing

[4] See `https://github.com/GFNOrg/GFlowNet-EM/` for the accompanying code.

on the forward logits. Here x is sampled from the empirical distribution and z is sampled from the forward policy π_ϕ. Additionally, I apply a sleep weight γ to strengthen the gradient. Formally:

$$\nabla_\phi \mathcal{L}_{\text{Replay}} = \mathbb{E}_{x\sim X, z\sim\pi_\phi(z|x)} \left[-\gamma \cdot \log \pi_\phi(\tau|x,z)\right] \tag{10}$$

If a correct solution has been found during the E-step, I immediately let the model train on these trajectories of correct solutions so as to consolidate these. After the E-step I again train the model on a set (in the mathematical sense, meaning no duplicates) of all the correct solutions, so that it doesn't forget solutions to other tasks. Replay is applied stochastically, given the hyperparameter `replay_prob`.

Fantasy During Fantasy I train the model on programs constructed during the E-step and run tasks from the empirical distribution through those programs to create correct task-program pairs (x, z). Then, similarly to the methodology of Replay, I train the model on these pairs. Formally:

$$\nabla_\phi \mathcal{L}_{\text{Fantasy}} = \mathbb{E}_{x\sim X, z\sim\pi_\phi(z|x)} \left[-\gamma \cdot \log \pi_\phi(\tau|x,z)\right] \tag{11}$$

Fantasy is also applied stochastically, given the hyperparameter `fantasy_prob`.

filtering out programs that produce None, constants, etc.

Fantasy on incorrectly proposed programs or randomly generated programs from the generative model lets the model learn which task-program pairs make sense, while fantasy on correct programs lets the model generalize solutions to different input-output pairs.

2.1.5 Optimization Techniques

Hu et al. propose a few optimization techniques which proved to be useful [30]. I adapted these techniques and will describe them in the following.

E-step Loss Thresholding Rather than training the GFlowNet to a loss of zero after each M-step, we can apply a linearly decreasing moving average loss δ as a threshold to trigger the M-step using the hyperparameter α, to save computational cost. Here I use the recursive formula:

$$\delta = \alpha \cdot \mathcal{L}_{TB} + (1-\alpha)\cdot\delta \tag{12}$$

Exploration Since we want to find many modes in the E-step and want to avoid getting stuck in local optima, several exploration techniques can be employed. The hyperparameter $\{\beta|\beta \in \mathbb{R}_{[0,1]}\}$ can be used to exponentiate the forward policy: $\pi_\theta(s_{t+1}|s_t)^\beta$. Moreover, ϵ-uniform sampling can be used to the deter the model from repeating known routes by mixing the predicted logits with a uniform distribution. ϵ is chosen to be $\{\epsilon|\epsilon \in \mathbb{R}_{[0,1]}\}$.

Algorithm 1 FlowCoder

Require: Data X, generative model with parameters θ, forward policy with parameters ϕ, optimization and exploration hyperparameters, threshold α

repeat
  Sample $x \sim X$
  Sample $z \sim \pi_\phi(z|x)$
  E-step: gradient update on ϕ with $\nabla_\phi \mathcal{L}_{TB}$
  if $r \in (0,1) <$ `replay_prob` **then**
    gradient update on ϕ with $\nabla_\phi \mathcal{L}_{\text{Replay}}$
  end if
  if $r \in (0,1) <$ `fantasy_prob` **then**
    gradient update on ϕ with $\nabla_\phi \mathcal{L}_{\text{Fantasy}}$
  end if
  if $\mathcal{L} < \alpha$ **then**
    Sample $z \sim \pi_\phi(z|x)$
    M-step: gradient update on θ with $\nabla_\theta[-\log p_\theta(x|z)p_\theta(z)]$
  end if
until some convergence condition

explain algorithm, check for additional hyperparameters, etc. should the gradient update of mstep be just the reward?

2.2 Model Architecture

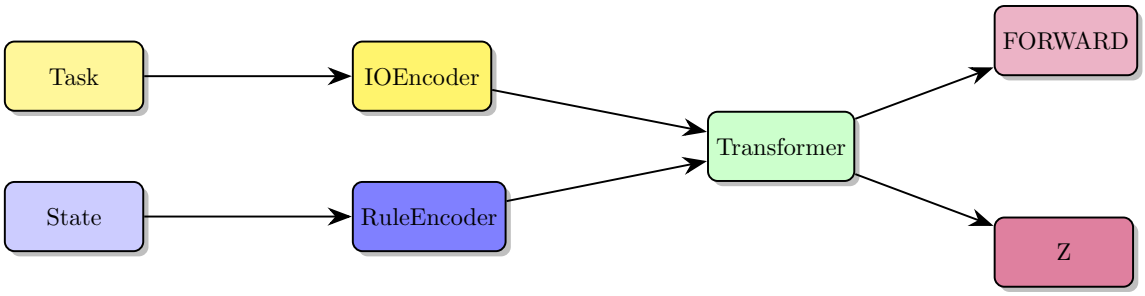

Figure 5: Insert explanation + maybe change task to an actual task; same with state; maybe also show the forward and Z output better.

this looks like its one model show that the transformer output is a state representation. the GFlowNet and Z takes it and produces logits.

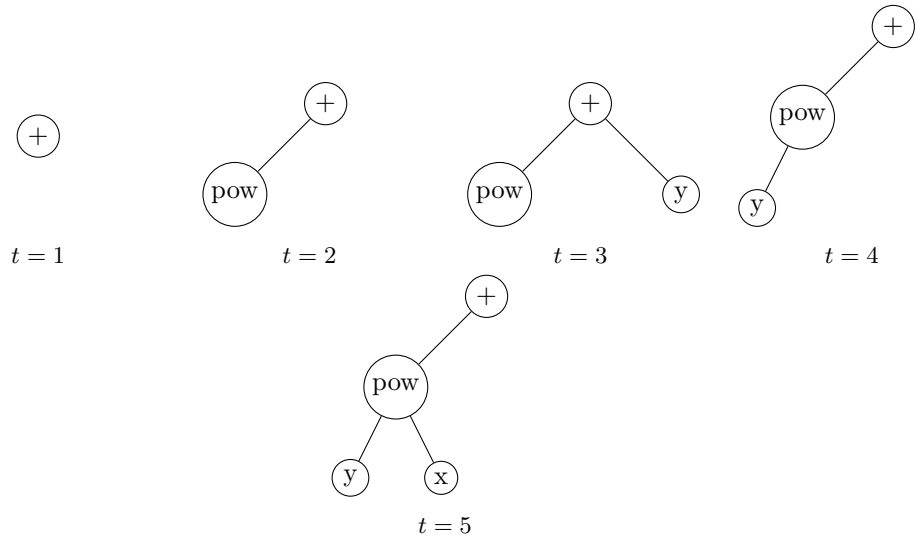

Figure 7: Creation of a five node tree using the grow initialization method with a maximum depth of 2, using terminal set T and function set F defined earlier, (t = time).

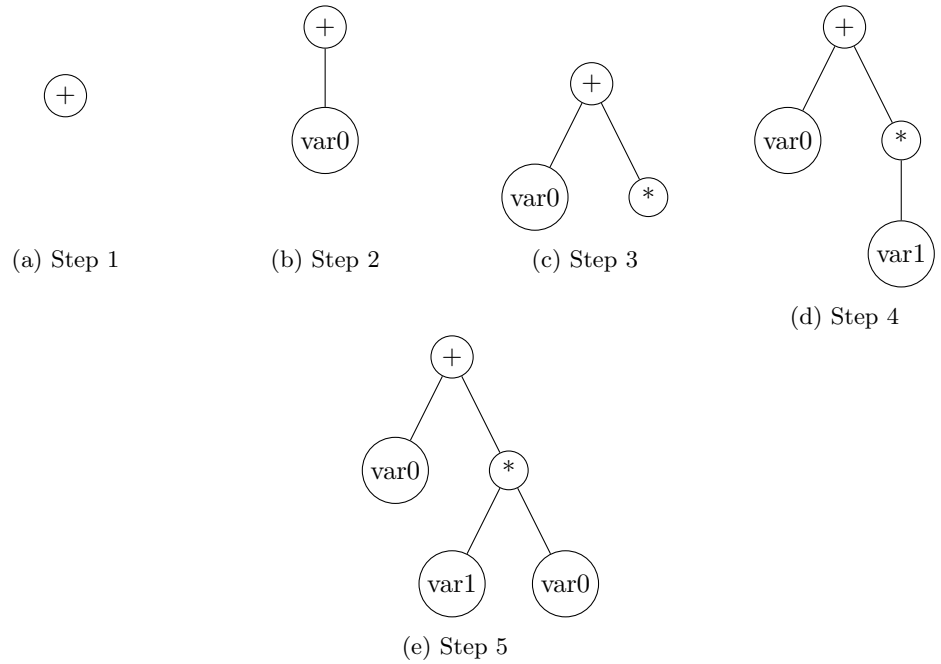

Figure 6: Incremental construction of the abstract syntax tree (AST) for `var0 + var1 * var0`.

See table 3 for the exact parameterization of the model.

2.2.1 Generative Model

To embed programs effectively within a neural network, it is essential to represent them in a format that is compatible with the neural network's architecture. Abstract syntax trees, which are inherently 2-dimensional and include information of parent, children and sibling nodes, should be translated into a neural network format while ensuring the preservation of this valuable structural information. Various concepts have emerged, including using Graph Neural Networks (GNNs), e.g.

see [31, 32, 33, 34]. Others proposed special Tree-Transformers which are able to encode ASTs, see e.g. [35, 36]. He et al. find that standard Transformers achieve similar results to Transformers where tree position information is explicitly encoded [37]. This is presumably because of the positional encoding, which is a way of adding fixed sinusoidal functions with different frequencies and phases to the embeddings of tokens in a sequence, enabling the neural network to discern the position of each tokens through these distinctive patterns; and more importantly, because of the fundamental component of the Transformer, which is the self-attention mechanism, formalized as [16]:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{13}$$

This mechanism, through the query (Q), key (K), and value (V) matrices, allows the model to dynamically assign importance to different parts of the input sequence. The scaling factor $\sqrt{d_k}$ normalizes the dot product, aiding in stabilizing gradients during training. Moreover, the Transformer employs Multi-Head Attention, enabling the model to concurrently process information from different representation subspaces, enhancing its capability to capture diverse features. Therefore, I decided to use the standard Transformer architecture, which takes a 1-dimensional sequence as an input, meaning the AST has to be linearized.

Rather than embedding the primitives of the DSL to construct ASTs, I embed the already preprocessed rules of the context-free grammar, created in conjunction with the syntactic constraints, thus essentially predicting edges rather than nodes. The array of CFG rules has to be converted back to AST format for evaluation, which can be easily done within the DeepSynth framework. In a plausible model of cognition, we don't possess an explicit representation of the CFG but rather infer it. However, in the case of DeepSynth, it generates a more extensive CFG instead of conducting type inference, which is the approach employed in DreamCoder. Consequently, I must perform a lookup to identify the syntactically permitted rules, thereby filtering out those that do not conform to the syntax. Naturally, in principle, I could allow the model to generate syntactically incorrect programs and assign them a reward of 0 during evaluation, but to expedite the learning process, I refrain from doing so.

The RuleEncoder begins by collecting rules, which are pairs of non-terminals and corresponding program actions, from the CFG and adds special tokens such as 'PAD' (padding), 'START' (sequence start), and 'STOP' (sequence end). Each rule is passed through a PyTorch embedding layer. During the forward pass, batches of state sequences (each state sequence, or trajectory τ, representing a series of CFG rules) are processed. To ensure uniformity across different sequences within a batch, padding is applied.

The IOEncoder is tasked with encoding input-output (IO) pairs into continuous vector representations. Each input-output pair is tokenized using a predefined lexicon (a list of symbols representing the possible range of inputs and outputs), which includes special tokens such as 'PAD' (padding), 'IN' (input start), and 'OUT' (output start). This tokenization is critical for distinguishing between different parts of the input-output pairs. The tokenized input-output pairs are concatenated into a single sequence, with the 'IN' and 'OUT' tokens demarcating the transition from input to output. Padding is then applied to ensure that all sequences have the same length, aligning them to the maximum allowed size determined by `n_examples_max` (the maximum number of input examples that can be processed) and `size_max` (the maximum size of elements in a list). These parameters have been adapted from DeepSynth. The

padded sequences are passed through a PyTorch embedding layer. This embedding is crucial for capturing the semantic relationships between different tokens.

The Transformer initializes with the IOEncoder and the RuleEncoder. Positional Encoding is applied to the output of both encoders. This step is vital as it adds information about the sequence order to the model, allowing the Transformer to interpret the sequence data effectively. The Transformer employs two types of masks: padding masks for IO sequences and square subsequent masks for state sequences. The padding mask ensures that the model does not process padding tokens, while the square subsequent mask prevents positions from attending to subsequent positions, maintaining the autoregressive property in the generation process.

2.2.2 Forward Policy

At each step, the forward policy takes the Transformer output as an encoded state and predicts log probabilities over the CFG rules, after which I apply softmax to get a distribution between 0 and 1 and sample the next action. The forward policy is implemented as a Multi-Layer Perceptron (MLP) and predicts forward logits from the Transformer's output, guiding the generative process.

2.2.3 Partition Function

The partition function Z_θ serves as a normalizing factor, ensuring that the probabilities generated by the model are well-calibrated and interpretable. It is implemented as a MLP layered on top of the Transformer output.

2.2.4 Sampling Programs

In the process of constructing an Abstract Syntax Tree, there exists flexibility in the order of expansion. Nodes can be expanded in a depth-first, breadth-first, etc. manner, or for instance, one can adopt a bottom-up approach, wherein terminal nodes are predicted initially and subsequently connected in a progressive manner. This approach offers the advantage of enabling the evaluation of partial expressions, which, in turn, can serve to inform the model and enhance computational efficiency, albeit at the expense of increased memory requirements. Alternatively, we may consider employing a model akin to the forward policy, predicting the subsequent node (or edge) to be expanded. However, I chose to sample the actions for the tree construction in a depth-first manner. I did this for two reasons. First, for simplicity, to not overcomplicate the model, and second so that the AST when linearized, always has the same order, potentially giving the Transformer a useful inductive bias.

Specifically, the state of each program in the batch is initialized with a 'START' token, representing the initial state of the program generation process. A frontier, implemented as a queue, is initialized for each program to manage the sequence of non-terminals that need to be expanded. This method has been adapted from DeepSynth to fit with the FlowCoder framework and mechanisms. The core of the method is a loop that continues until all frontiers are empty, indicating that all programs in the batch have been fully generated. Within this loop, the model computes logits and partition functions. As discussed in section 2.1.5, the sampling method includes an exploration mechanism, where with a probability ϵ, uniform sampling is used instead of the model's logits. This exploration is crucial for introducing variability and avoiding local optima in the

generation process. Additionally, tempering is applied to logits using a factor β, modulating the sharpness of the probability distribution used for sampling. For each program in the batch, the method iteratively samples a rule based on the current non-terminal and updates the program's state and cumulative logits. This process involves creating a mask to block invalid actions, applying the mask to logits, and sampling an action (rule) based on the masked logits. Each sampled rule is decomposed into a non-terminal and program component, updating the program's current state and expanding the frontier with the rule's arguments. Once all frontiers are empty, the final programs are reconstructed from their compressed representations, using methods provided by DeepSynth.

Refer to program construction diagram

detailed algorithm of replay and fantasy

2.2.5 Reward

In order to train the model, we need to operationalize a reward function. Various approaches exist for this purpose, the simplest being a binary reward, as employed in DreamCoder. Does the output of the program match the actual output or not? A binary reward however, is not very informative. A reward that provides a gradient is much more useful. Bengio et al. propose an Energy-Based Model (EBM), wherein the model learns to associate favorable outcomes with low energy states and unfavorable outcomes with high energy states [28]. Depending on the domain, this may be useful but in my experiments, working in the list editing domain, however, I found that it introduced unnecessary complexity to the model. Instead, I am using the Levenshtein edit distance, using the `Levenshtein` package [5], which is a measure of the similarity between two strings. Specifically, it quantifies the minimum number of single-character edits (i.e., insertions, deletions, or substitutions) required to transform one string into another. See appendix 5 for an in depth formalization of the metric. Since the Levenshtein distance returns a discrete value, I normalize it over the maximum length of the sequences, and since there may be more than one example per task, I average it over all examples. Moreover, I apply a maximum reward parameter, to scale a correct solution up and give the model a stronger gradient. In my experiments I used a maximum reward of 10.

2.3 Design

In the following section I will elaborate on the exact training and test methods I used, including hyperparameters. The tasks I am using are input-output relations in the list editing domain, originally from DreamCoder, see Table 2 for examples. These were filtered given the syntactic constraints (e.g. type, lexicon, etc.) and provided by DeepSynth. In their paper, Fijalkow et al. discuss that some of the tasks are impossible solve given the DSL, so I additionally filtered those out. In my experiments I end up with 95 tasks. Tasks can be of a similar variety (e.g. `add-k with k=1` and `add-k with k=2` belong to the same group). There are 25 such task groups in total. Each task can have between 1 and 15 examples. A task is considered solved if a program solves all examples of the task [6]. The DSL is essentially a dictionary of primitive types and semantics in a typed λ-calculus, including functions like `index`, `car`, `append` as well as numerical functions like

[5] `https://github.com/maxbachmann/Levenshtein`
[6] All tasks can be found in my GitHub repository.

`+`, `*`, `mod`, `is-prime`, etc., all written in Python (see appendix 1). Table 1 describes the syntactical constraints used in the experiment, which were mostly adapted from DeepSynth and DreamCoder.

Parameter	**Value**	**Description**
`type`	`list(int)` $\rightarrow$ `list(int)`	The input as well as the output should be a list of integers, so the CFG should reflect that, filtering rules that do not conform to that criterion.
`lexicon`	$[-30, 30] \in \mathbb{Z}$	The lexicon is a uniform distribution of integers and in the range from -30 to 30
`maximum argument number`	1	This is the maximum number of arguments a function could have.
`size max`	10	The maximum number of values in a list
`max number of examples`	15	The maximum number of examples per task

Table 1: Syntactical Constraints

All experiments were trained in random order with a batch size of 4 where each task in the batch is the same. This was done to avoid a credit assignment problem while increasing efficiency. Since the forward logits are summed up and averaged over for all trajectories, training the model at multiple task at once might have confused which trajectory was rewarding. However, batching, especially when training on a GPU has computational benefits. Furthermore, each task is trained for 5 epochs with 2.000 E-steps and 2.000 M-steps in each epoch. The moving average threshold (see equation (12)) δ was initialized with 150 and linearly decreases using the hyperparameter α set to 0.3. The exploration parameters β and ϵ were set to 0.7 and 0.3, respectively. The sleep weight γ was set to 10 on all experiments. The learning rates for the generative model as well as for the forward policy were set to 0.0001. During inference I ran the model for 100 steps. Since the batch size is 4, this creates 400 programs per task. A table of all hyperparameters can be found in Table 2 of appendix 2. The experiments were run on a single NVIDIA Tesla T4 GPU. Throughout the five epochs, each consisting of 2.000 E- and M-steps with a batch size of four, the model generated approximately 80.000 programs per task.

Task	**Input**	**Output**
`remove gt 2`	`[1,2,7,5,1]`	`[1,2,1]`
`caesar-cipher-k-modulo-n with k=5 and n=4`	`[2, 2, 0, 1, 2, 3, 3]`	`[3, 3, 1, 2, 3, 0, 0]`
`prepend-index-k with k=3`	`[15, 12, 9, 14, 7, 9]`	`[9, 15, 12, 9, 14, 7, 9]`
`add-k with k=4`	`[16, 10, 7, 12, 13, 3]`	`[20, 14, 11, 16, 17, 7]`
`append-k with k=2`	`[1, 5, 15]`	`[1, 5, 15, 2]`

Table 2: Task Examples

Task	Program
remove gt 2	(filter (gt? 3) var0)
caesar-cipher-k-modulo-n with k=5 and n=4	(map (mod 4) (map (+ 5) var0))
prepend-index-k with k=3	(cons (index 2 var0) var0)
add-k with k=4	(map (+ 4) var0)
append-k with k=2	(append 2 var0)

Table 3: Examples of tasks and programs solving the tasks.

3 Results

3.1 Experiment 1

This experiment tested the model's ability to generalize to unseen tasks. I set the program depth to 4 and trained on a random selection of only 17 out of 95 tasks. I set both the `replay_prob` as well as the `fantasy_prob` parameters to 1, increasing the sleep frequency. Replay and fantasy are applied within each E-step, whenever a correct solution is found as well as after each E-step. All correct task-program pairs are saved and trained on after each E-step. The training took approximately one week. The model solved 15 tasks during training and 33 during inference. Notably, 8 tasks solved in training were not solved during inference, suggesting the model occasionally lost correct solutions found during training. During inference, the model therefore solved 26 tasks it had not seen before. Out of these, 7 were from previously unobserved task groups, demonstrating the model's capacity for generalization.

3.2 Experiment 2

In this experiment I want to examine the model's frequency of resampling, whether the model proposes varied solutions, and analyse its efficiency. Moreover, I analyze whether the EM cycles prove to be useful.

In this experiment I trained the model on a random 50/50 train-test split, which took about 3 days. All programs can be solved with a maximum program depth of 3, so I limited the model to that depth. This was merely done for efficiency. Moreover, `replay_prob` and `fantasy_prob` were set to 0.3 and 1, respectively, and only a set (meaning no duplicates) of correct task-program pairs were saved and trained on, limiting the model's exposure to already correctly solved tasks [see sec for discussion on sleep]. The model successfully solved 33 out of the 48 tasks it was trained on. Although the model was trained on half of the tasks in the dataset, it does not generalize as well as FlowCoder in the first experiment. Sleep seems to be a crucial aspect of the algorithm.

As depicted in Figure 8, an average of about 11.000 unique programs were created per task, indicating that approximately one-eighth of the programs were resampled, while the rest were distinct. This data also reveals certain task groups that posed more significant challenges than others. For instance, none of the tasks in the `caesar-cipher-k-modulo-n` group were solved, whereas all tasks in the `prepend-index-k` or `mult-k` groups were successfully completed. Refer to Tables 2 and 3 for examples and correct solutions, respectively.

During inference, the model attempted to solve all 95 tasks sequentially, producing 400 programs in about 100 steps, taking roughly 30 seconds for each task. FlowCoder solved 42 tasks, of which 13 were unseen during training, as shown in Figure 9. 11 of these tasks belong to groups where at least one task was solved during training, e.g. the model was trained on (and solved) tasks `append-k` with `k=0, k=2, k=3` and during inference the model solved tasks `append-k` with `k=1, k=4, k=5`, which were previously unseen. 2 tasks were solved during inference without any precedent of tasks of a similar task group being seen in training. Additionally, all tasks solved exclusively during inference had not been exposed to the model in the training phase. The distribution of unique solutions also reveals that multiple tasks, irrespective of their exposure during training, exhibited a range of solutions. This diversity in solutions reflects the model's capacity to explore the solution space comprehensively, aligning with our goal of not merely finding a point estimate but understanding the entire posterior distribution of solutions for each task.

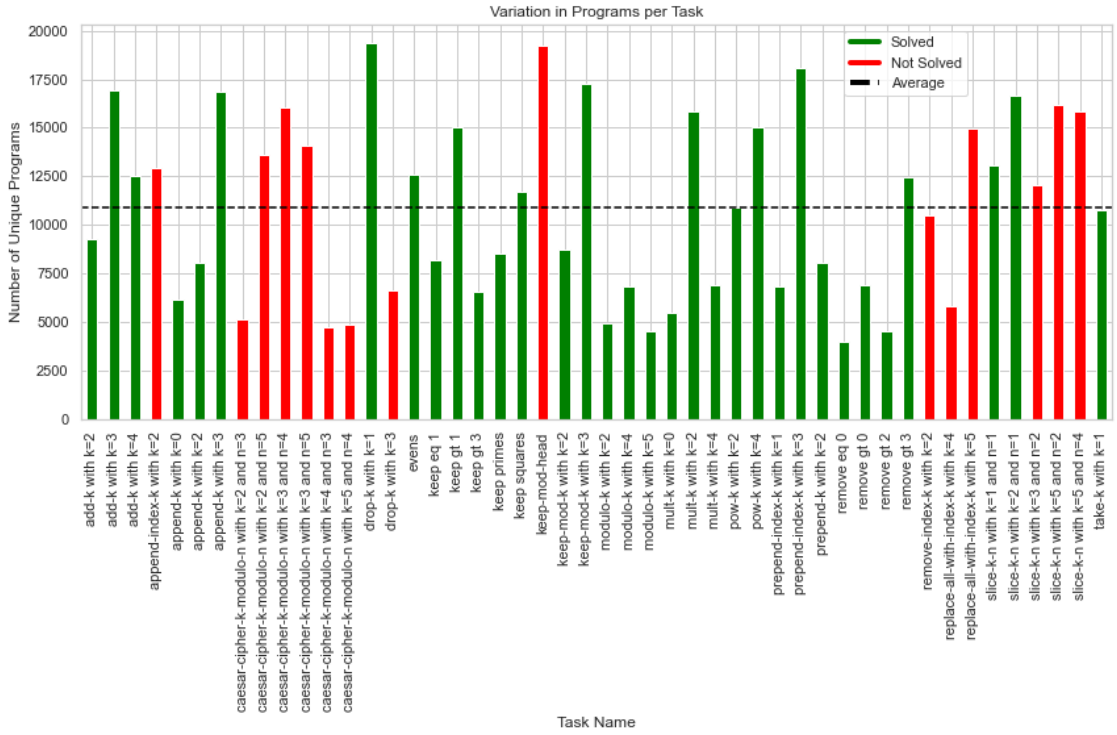

Figure 8: Bar plot of unique programs created per task. The sorted tasks the model has been trained are on the x-axis and the number of unique programs that have been created per task are on the y-axis. Bars of tasks that have been solved are coloured green and unsolved tasks are coloured red. The black dotted line demarcates the average number of uniquely created programs.

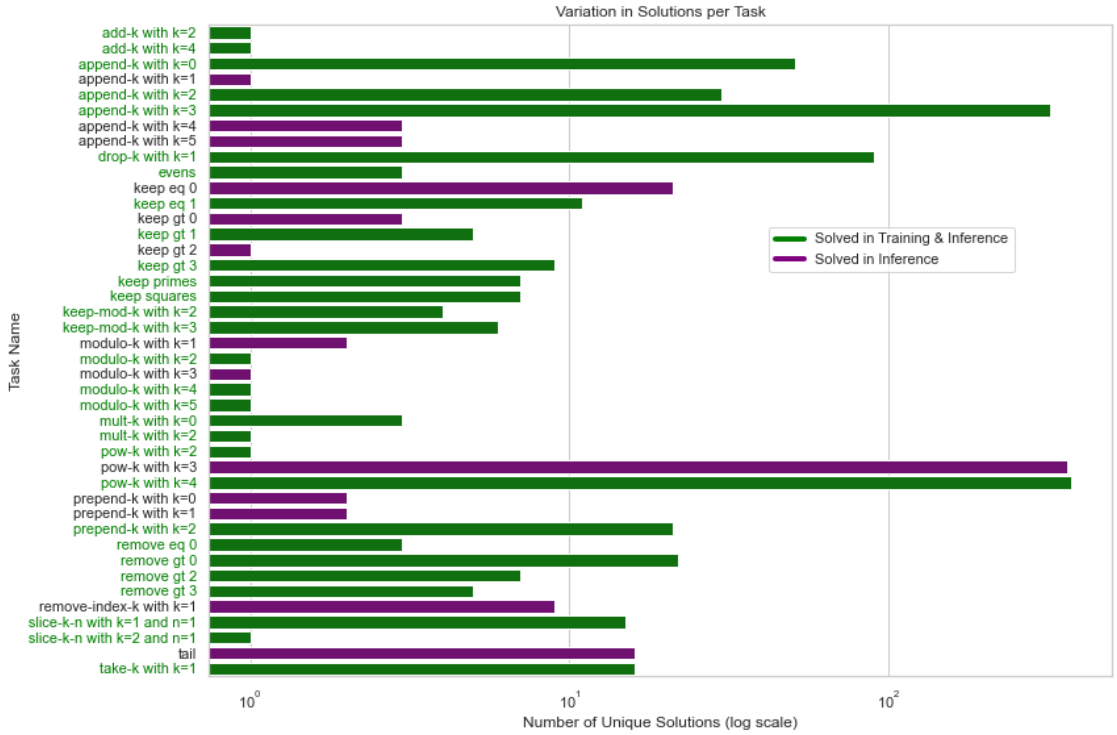

Figure 9: Distribution of unique solutions per task during inference, displayed in a log-scaled bar chart. The y-axis lists the task names, while the x-axis quantifies the number of unique solutions. The chart reveals that 13 tasks not solved in training were resolved during inference, with 11 belonging to groups with at least one task previously solved in training. Only solved tasks are shown.

Task	Solution

`append-k with k=2`	`(append 2 var0)`
	`(append (mod 4 2) var0)`
	`(append (min 4 2) var0)`
	`(append (mod 5 2) var0)`

Table 4: Multiple found solutions.

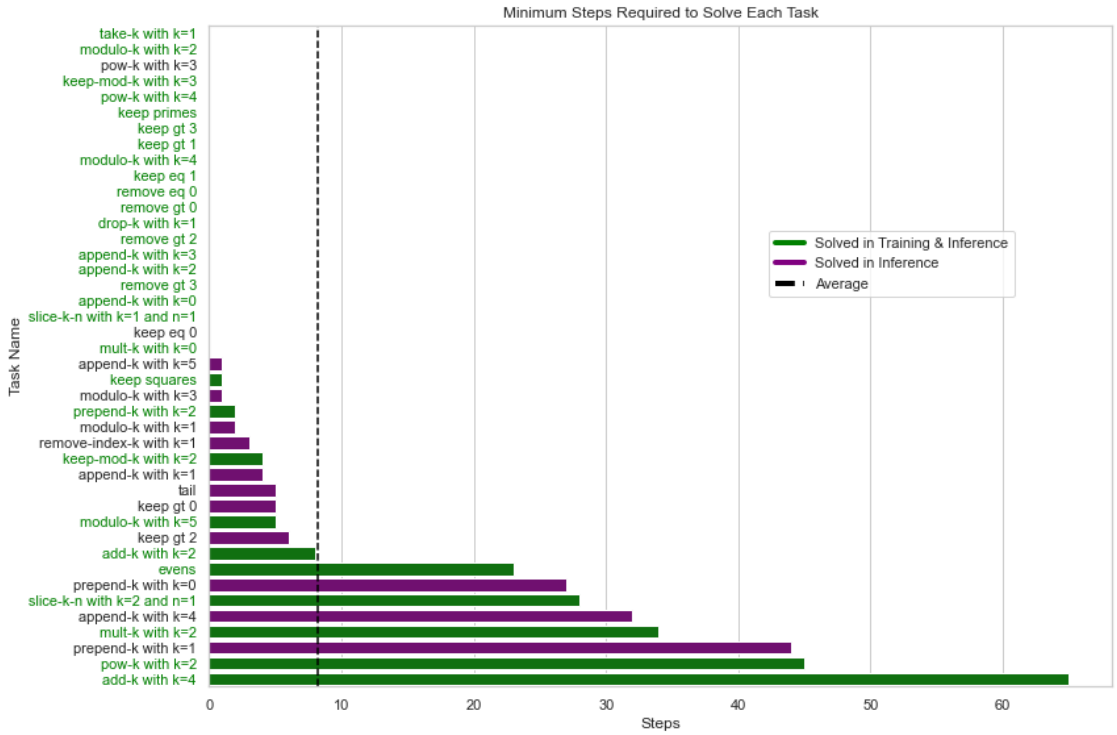

Figure 10: Analysis of the minimum number of steps required to solve tasks during inference. The x-axis represents the number of steps, and the y-axis lists the task names, sorted by the number of steps taken to find a solution. The dotted line indicates the average number of steps needed. Tasks solved both during training and inference are highlighted in green, whereas tasks exclusively solved during inference are in purple. Only solved tasks are shown.

In table 4 we can see an example of varied solutions to a task. The model has found the shortest solution but also alternate solutions of the same task.

On average, FlowCoder efficiently solves a task within approximately 8 steps. Notably, as illustrated in Figure 10, around half of the tasks are successfully resolved on the initial attempt. The model is able to rapidly solve many tasks that it had not encountered previously, often requiring fewer steps than the average. This suggests that when trained to convergence, the model may act as an efficient sampler.

An examination of the model's improvement across consecutive Expectation-Maximization (E-M) cycles is shown in Figure 11. The plot shows a clear upward trend in the average cumulative number of solutions, suggesting progressive improvement in both the forward policy and the generative model. The results of experiment 1 3.1 are similar. As expected, the generative model during the M-step performs better than the GFlowNet during the E-step, since the E-step includes exploration while the M-step does not.

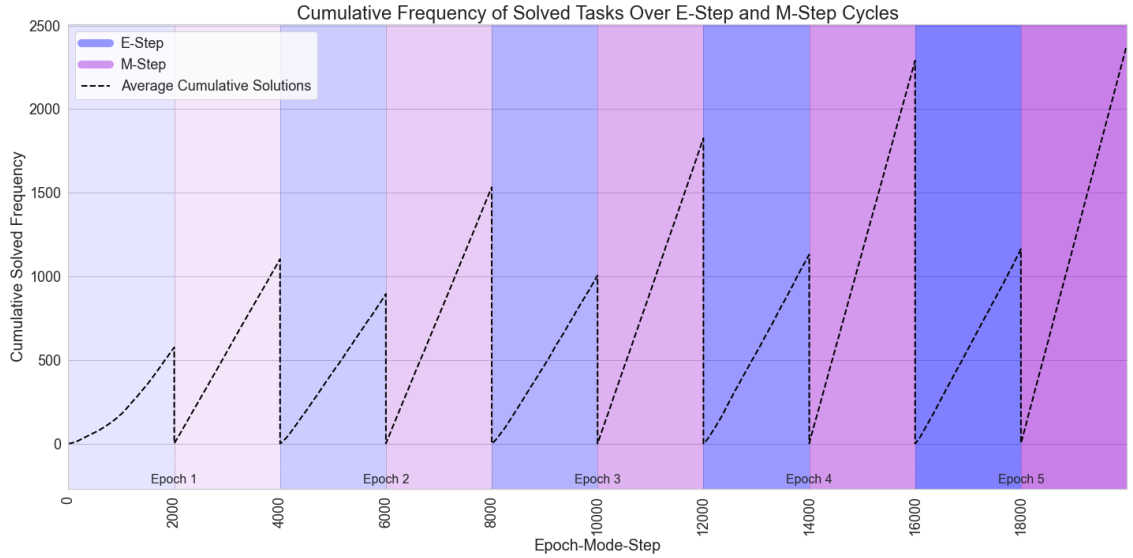

Figure 11: The plot displays the cumulative number of tasks solved (y-axis) against the number of steps (x-axis). Each step represents an iteration in the E-M cycle. The initial 2.000 steps correspond to the first E-step, marked with a blue background, followed by the first M-step spanning the next 2.000 steps (up to step 4.000), distinguished by a purple background. This pattern constitutes one complete epoch. The graph includes a dotted line representing the average number of tasks solved over all epochs, offering a benchmark for comparison. Furthermore, the intensity of the color hue in the plot encodes the temporal sequence of the epochs: brighter bars on the left signify earlier epochs (the first epoch), with the hue gradually darkening towards the right, culminating in the fifth and final epoch.

4 Discussion

In this section I will summarize key findings, and address the aims of the research.

what do we want from the model? - it should not resample too often, about an eighth (this can be improved). of course we are counting the correct solutions too, over and over - it should be fast (when converged, it's immediate) (however everythong above depth 4 was critical) - it should generalize OOD (it solves previously unseen tasks within and outside of task group) - sleep is crucial

In the following section, I will compare FlowCoder to the concepts introduced in section [introduction or more specific 1.1, 1.2, etc.]

4.1 Experimental Results

- since the model solves most tasks within groups, it seeps to suggest that it learns relationships between groups (program space).

The experiments of section 3 revealed that FlowCoder can be trained as an adequate program synthesizer. First, we found that the model solves more tasks in inference than in training, including tasks of groups which had previously been unseen, suggesting some out-of-distribution generalization capabilities. Moreover, the model produces varied solutions, not only converging on a point estimate, the most likely program, but even in inference proposes multiple programs solving the tasks, suggesting that it has found a multi-modal distribution. In section [x] we have discussed why a multitude of representations are useful. In the context of program synthesis however, we

must not conflate semantics with syntax [see discussion]. As shown in table 4, many inefficient alternate solutions to a task were found. We would actually like to avoid programs that add 0, multiply by 1, etc. Breaking syntactic symmetries can be done by optimising for minimum description length and including an abstraction phase as is done in DreamCoder. The model is able to sample varied correct solutions on the first step, suggesting that when we make the investment of training the model to convergence, we reap the benefits in the form of one-shot inference. The model's performance steadily increases with alternating E-M steps, suggesting that it is beneficial to train the generative model separately from the forward policy and having the two models bootstrap each other. The generative model proposes better representations of the program space while the forward policy selects better actions, proportional to the posterior distribution, given the task at hand.

Many tasks have not been solved, but it looks like if the model had been kept training, it could converge on those too. moreover, we need to adjust resampling since some tasks have not been solved but have been resampled many times.

- finding a balance between - exploration (beta, epsilon, fantasy) to find many modes, E-step - exploitation (replay) converging on modes, not forgetting, M-step and exploitation (exploration to find many modes) but also exploitation

Fijalkow et al. compare different search strategies and show that methods that do not use a machine-learned PCFG (e.g. depth-first search (DFS)) barely solve any tasks (max. 5) in over 1.5 hours [24]. [this should be in intro]

- More tasks were solved in inference. since im going chronologically through the training set, the model doesn't see the first task ever again. this would be beneficial. minibatch sampling.
- there are many other training strategies that could be employed.
- symmetry breaking. we want to avoid if True, or $+0$ or $*1$, etc.
- benefit over dreamcoder etc. we can ask questions about partial programs. we are embedding programs as well as tasks, we can input a partial program into the model and ask what the next best step is. We can also train from partial tasks etc. making this method more dynamic. [to introduction?]

Sampling strategy ASTs can be constructed in various orders. Ideally, we could let the GFlowNet predict which node to expand. However, this would have meant creating an additional model as a node predictor, which increases FlowCoder's complexity. Moreover, since ASTs are inherently two dimensional and I am using a transformer to encode CFG rules as a representation of a program, ASTs have to be linearized. I decided to expand the tree in order of depth-first, both to eliminate the need for another model and also to give the transformer the input in consistent order, making learning easier.

- bottom-up construction could be beneficial as nodes can already be evaluated - construction order - include diagram

DreamCoder enforces parsimonious programs during abstraction. Common patterns are refactored, shortening the programs. Deepsynth enforces programs of minimum description length by optimising for the expected number of programs before finding a correct solution. This works, if programs are considered in order of increasing length. Both of these methods however optimise

for point estimates, i.e. the most likely program for a given task. Instead, I want to learn the posterior distribution, i.e. all solutions to a task, which of course is much harder. I do this, first, because if we see that the model finds the posterior, it can always be simplified to MAP. Second, given the analogy of humans understanding the world via program-like concepts, we know that we can represent the world in various ways and we can find multiple solutions to a problem. Consider the following example: the task

I constrain the program depth to 3, which is enough to solve most programs and is not too computationally heavy. Since most tasks can be solved by short programs, Despite the minimum description length being a factor which may be necessary, I relax this condition

4.2 Comparison to DreamCoder

4.3 Speed

It is somewhat difficult to compare the time efficiency of my method with other approaches for various reasons. Both DeepSynth and DreamCoder predict weights for the PCFG and then parallelize search processes on multiple CPUs. DreamCoder train their model for about a day on 20-100 CPUs and it takes around 10 wake-sleep cycles to converge in the list domain. I instead am working on one GPU (with possible interruptions from other users on the cluster). DreamCoder shows that the refactoring algorithm used in abstraction is crucial in the list processing tasks, which I did not implement. DeepSynth does not include model query time in their computations. I am training my model on a filtered set of the original DreamCoder dataset, making it difficult to compare with DreamCoder and some of the DeepSynth experiments.

[Moreover, I did not work too much on time optimizations and still it seems that it isnt too slow in a rough comparison with those other methods.]

4.4 Scaling the Model

check this, check what is necessary

The time complexity for encoding an IO pair is $\mathcal{O}(n)$, where n is the length of the IO pair.

Similarly, the RuleEncoder's time complexity is $\mathcal{O}(n)$ where n is the number of rules.

In each step of the trajectory, I embed the task and the current state, and use the transformer to construct the trajectory until completion.

The transformer's complexity is $O(num\_layers \times (d_{model}^2 \times n_{seq} + n_{seq}^2 \times d_{model}))$, where num_layers is the number of layers, d_{model} is the model dimension, and n_{seq} is the sequence length.

The linear layers within the forward policy π_θ and partition function Z_θ have a complexity of $O(d_{model}^2)$ due to the matrix multiplication involved.

The overall time complexity of a forward pass through FlowCoder is dominated by the transformer's complexity, which is $O(num\_layers \times (d_{model}^2 \times n_{seq} + n_{seq}^2 \times d_{model}))$.

Since I have linearized the construction procedure, I reconstruct the list back to AST format and then evaluate the tree to get the program output.

The reconstruction and evaluation are both recursive processes that depend on the depth of the tree and number of arguments in each partial program. The worst-case complexity of program reconstruction, as well as the evaluation can be approximated as $O(k^d)$, where d is the depth of the recursion, which corresponds to the length of the program list, and k is the average number of

arguments for each function in the program. In my experiments only one argument per function is allowed.

[sec x to see how we can mitigate this (better program representation, construction, evaluation (instead of reconstruction, reformating etc.))] In general i want to show that the model scales.

Subsequently, I compute the reward.

The Levenshtein edit distance algorithm, when implemented using dynamic programming, has a time complexity of $\mathcal{O}(m \times n)$, where m and n are the lengths of the two input strings.

4.4.1 Scaling

- Scaling the model (How does the model scale with larger CFGs?)
- We need to show that with abstraction, we could theoretically scale
- Scaling neural search paper
- how would it work using GFNs
- how would it work if we use abstraction like in DC? DC shows it is scalable.
- this needs to be related to increasing number of primitives, maybe look at dreamcoders complexity. we save complexity by not having to search, and also space, since we dont have to save anything, we just need to train a phat model

The scalability of the model depends on various factors. First, we have seen that GFN can deal with the marginalization term and is an instance of amortized bayesian inference. This means that we can capitalise on seen data to generalise to many modes even in high-dimensional spaces. [show that amortized inference scales] The intention of this study is to show that the general method of using GFNs for program synthesis is useful and not to optimise it for efficiency and scalability. Many adjustments can be made to accommodate for scaling. program representation, choosing the generative and forward (and possibly backward) policy models, parallelizing tasks, different training strategies most importantly, as discussed in [sec DreamCoder], the abstraction phase is crucial to narrow the depth of the search tree. (and MDL). The end-goal is to have an increasing DSL as well. Scaling neural search paper all of the program construction/ reconstruction/ evaluation can be reduced.

the problem with scaling is marginalization. DC deals with this by marginalising over a beam of programs for each task. GFN deals with it by adhering to the flow objective. other amortized variational inference methods, ...

4.5 program representation

Other ideas to represent ASTs are GNNs, or Tree-Transformers [which were explored, but the computational complexity gets higher [source]. also distributed representations of aggregations are an option like in GFN-EM.], however, I settled on using a regular transformer, since it has been shown that it can learn tree structures implicitly, through positional encoding and also [source] show that the performance is similar. since we are representing trees, it might be beneficial to embed programs in hyperbolic space [see conceptual spaces]. There are other ways to embed the cfg, [e.g. see kim]. - 3-index bigram, .. compound PCFG including information of neighboring states, parents, childrewn, etc.

since we are representing the tree as an array of rules, so that we can use them with transformer, we need to reformat it back into a tree to evaluate it. i.e. we need to fisrt construct it and then go through it again for evaluation. this slows down the computation.

- heapsearch, evaluation, using sub-trees, etc. Another expansion is using a depth first search (DFS) [source] approach (note that this is not about searching through the CFG, but constructing the AST). Since we are essentially linearising a tree when giving it as input to the transformer, the transformer has additional order information, which in combination with positional encoding lets it learn better.

hyperbolic space

We could embed parent, child, etc. to make ean embedding of a rule more informed. i chose not to do it to see whether its enough (discuss in limitations,e tc.)

Syntax vs Semantics Semantics can be understood as data flow, e.g. partial evaluations could be used as part of the representation of the program. type information For a discussion on semantics and meaning see sec [meaning] In LLMs, we see that they l

4.6 Program Embeddings

programs are commonly represented as AST.

Does DC learn a program space (discuss their tsne) - hyperbolic

Graph Neural Networks seem like an interesting option to represent ASTs Allamanis et al. use Gated Graph Neural Networks to include local semantics in their program representations

Allamanis et al. use Gated Graph Neural Networks to represent both syntactic and semantic information by including data flow and type hierarchy signals [31]. The authors formalize how to turn ASTs to graphs.

Wang et al. propose a semantic program embedding, learnt from program execution traces [38].

Ibarz et al. present a generalist neural algorithmic learner by leveraging GNNs [34]. Zhang et al. propose a novel neural AST representation of source code [39].

Oliviera and Löh develop abstract syntax graphs (rather than trees) to preserve sharing and recursion within a DSL [40].

4.7 reward

- Reward should be learnt (Energy based model). Where does the brain get it from?
- Is Levenshtein a reasonable reward? think about computational complexity (Kolmogorov complexity)
- Reward depends on tasks
- Rewards can be based on utility or efficiency rather than accuracy, or on aesthetics or other determinants. This is a complex topic which should be investigated in the future.

4.8 Limitations

- Make the network choose where to expand (not DFS). Note that we are not going through the CFG DFS, but creating the AST with DFS (GNN, evaluation etc.)
- What should be the best order of solving tasks? should we try to solve a certain task for x amount of time and then move on to the next? curriculum learning?

- Better reward

4.9 Related Work

- apperception engine
- one language to rule them all
-

4.10 Future Work

- testing sleep, exploration, etc.
- Sub-trajectory balance
- Learning from dataset of good programs (akin to someone telling you a solution)?
- Train from middle of states
- Train backward policy, then we can also see finished states and predict trajectories that led to them
- caching evaluations, bottom-up
- we could let automate how long to train. if it is solving the task in 100 consecutive e steps and also in m stepts, it could also just move on.
- Hyberbolic space (relate to holarchies)

4.10.1 Comparison to other approaches

Previously, techniques in approximate Bayesian inference, have been applied to deal with the difficult marginalization term.

Variational Inference (VI) is a strategy that leverages a simpler distribution $q(z|x)$ to approximate a more complex target distribution. This is achieved by minimizing the Kullback-Leibler (KL) divergence, a similarity measure of probability distributions, between the true distribution and its approximation.

$$D_{KL}[q(z|x)||p(z|x)]$$

The approximation can be optimized by increasing the Evidence Lower Bound (ELBO) by:

$$\text{ELBO} = E_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)||p(z|x)) \tag{14}$$

Where $E_{q(z|x)}$ denotes the expectation under the recognition density q.

In the context of *amortized* VI, computation of the variational parameters ϕ is shared across multiple data points. Traditional VI might determine a unique set of variational parameters ϕ_i for each data point x_i, which is computationally intensive. Amortized VI, however, leverages a function (commonly a neural network) to compute the variational parameters ϕ for any data point x in a single pass, enhancing efficiency. In other words, we parameterize the recognition density.

this aligns with the free energy principle formulation [show that.] maximizing the evidence lower bound (ELBO), is equivalent to the negative free energy.

Amortized variational inference refers to the use of a parameterized function (e.g., a neural network) to approximate the posterior distribution in a variational Bayesian setting, where the parameters are learned once and can be used to infer the posterior for any given data point without retraining. It is "amortized" because the cost of learning the inference model is spread over all the data points it is used on.

GFlowNets relate to amortized variational inference in the sense that they learn a policy network that can generate samples for any given reward function without having to solve a new optimization problem for each sample. This is similar to how an amortized inference model can be applied to new data without additional optimization.

In both cases, the "amortization" allows for efficient inference or generation after the initial cost of learning the model. However, GFlowNets focus on learning to sample in proportion to a reward function, whereas amortized variational inference is concerned with approximating the posterior distribution of latent variables given observed data.

- Relation to MDPs, POMDPs
- Relation to Reinforcement learning
- Relation to MCMC sampling
- Relation to Variational inference.

variational approximation we need an ELBO but with GFNs we don't

A variety of techniques have emerged in the domain of program synthesis, [each with pros and cons].

Ullman et al. MCMC[source, definition] and Metropolis-Hastings[source, definition] [20]. [include pros and cons]

should the above be in the intro??

4.10.2 Biological Plausibility

- Why are/ aren't types plausible? Related to categories?
- CFG? where does it come from?
- primitives?
- Discuss Abduction, etc.
- of course a model that can create programs of depth 3 is somewhat underwhelming when compared to human cognition.

4.11 Conclusion

- The conclusion here is that GFN is a viable method for program synthesis which could replace models such as DC.
-

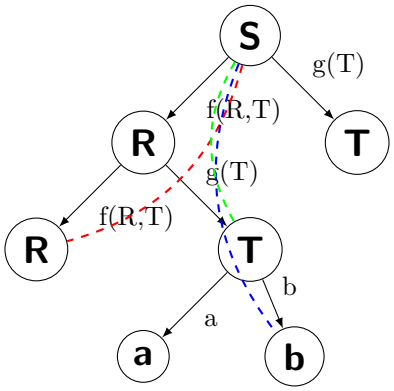

Figure 12: Illustration of the tree of leftmost derivations.

- Robustfill
- DeepCoder
- Etc.
- Paved the way in a coubple of different ways, etc.

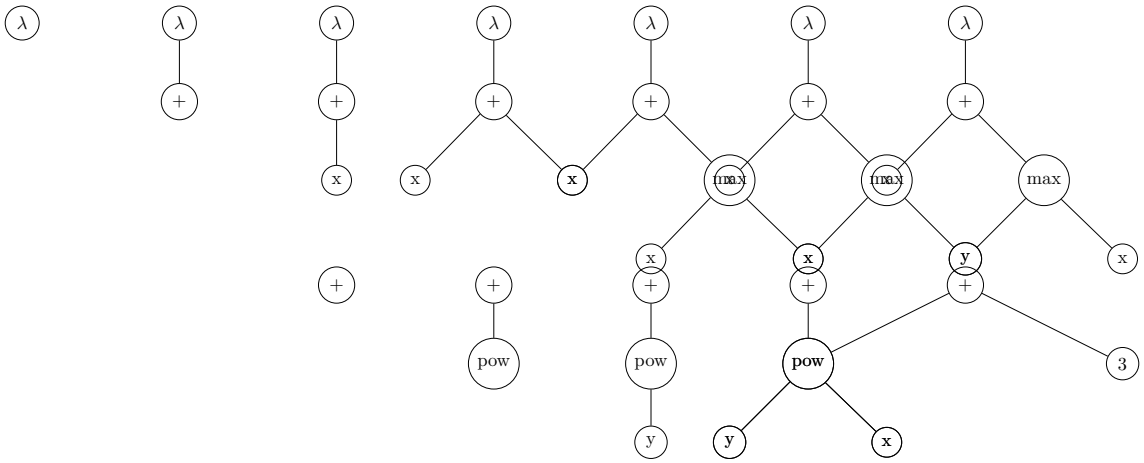

Figure 13: Creation of trees showing the construction process.

Chapter 1

Philosophical Ramifications

Here I want to discuss the paradigms assumed more generally, consider alternatives, and discuss the benefits and limitations of the approach and lastly consider the overarching elusive questions every cognitive scientist thinks about: who am I.

1 Language of Thought

The idea that the thoughts are constructed by some kind of Language of Thought (LOT), in order to understand and represent our reality has a long history. Gottfried Leibniz imagined a *Characteristica Universalis*, a formal language capable of expressing metaphysical concepts [41]. Jerry Fodor, among many other scholars, continued the development of this hypothesis and proposed that our cognitive processes are akin to computations involving a system of mental representations that can be composed into complex thoughts, akin to how sentences are formed from words according to the rules of grammar [42]. Fodor's theory implies an innate structure underlying human cognition, with systematic, rule-governed operations that manipulate symbols.

1.1 What is a Symbol? - Semiotics

Other things to clarify

- What is a symbol (also relate to semiotics but also in the cognitive sense. Something that relates to something else, a pointer. Think of memories that are stored in the hippocampus.) what about bioelectrical prepatterns? are all representations symbolic? How would symbols be implemented in the brain?
- Analytic vs. Geometric Solutions as an analogy to approximation vs symbol? imitating understanding vs actual and think about whether there is actually any difference. perhaps a limitation of the symbolic programming approach is that it is rigid and less flexible?
- Explain the different gradients of symbols. e.g. SPA is symbol-like. Part whole hierarchies and capsules are also kind of a symbol in that there is a central representation of a concept. I think its about centrality vs distributed representation and also about the capabilities of the symbols, i.e. does cognition work by only manipulating the symbols, rather that the representations themselves? What is that extra layer needed for then? Think of SPAs layered vectors, shallow vs deep understanding.

- If we argue that the LOT is a kind of a self-organising system, then what are the parts (the primitives) we construct (analogy to primes and how every natural number can be constructed using primes.) What are the primitives here? Do we have some axioms of ontology? Are they built in? I.e. taking the stance that we do not start off with a clean slate (Turing) (inductive biases?). Perhaps through evolution?
- Maybe talk about the idea that concepts might start off simple, i.e. starting with light and dark, warm and cold, then increasing the complexity of concepts, like Hubel & Wiesel.
- This kinda goes back that every sacchade is a query to the world. Werner Heisenberg "What we observe is not nature herself, but nature exposed to our method of questioning."
- Relate to poverty of the stimulus
- relate to piaget
- model vs model free (we dont just learn chess by watching, we either learn about or try to infer the rules.)

1.2 Semiotics

How can we classify concepts? Ferdinand de Saussure assumed the sign relation to be dyadic: there is the form of the sign, the signifier; and its meaning, the signified. C.S. Peirce thought the sign relation is triadic: there is a sign vehicle (form of the sign, the signifier); the sign object (the thing it refers to, the signified); and the interpretant (subjective interpretation). Peirce's signs can be deconstructed into three forms: Icons signs that signify by means of similarity between sign vehicle and sign object they have physical resemblance to the signified, like a picture of the signified Indices signs that signify by direct contiguity or causality between sign vehicle and sign object E.g. smoke to refer to fire Symbols signs that signify through a law or arbitrary social convention These are culturally learned, like the alphabet or the number "9"

About the grounding problem and semiotics: sure, initially, the concept needs to point to something in base reality but once the concept is formed, it can detach from that pointer, it can then be modulated, composed, recombined, etc. to create another concept or be part of a new concept. That way, the network of concepts does not need to be grounded anymore. It is useful though for the concepts to map to the world in some way. This is useful, not accurate. I suspect that the new concepts will then go through some neural darwinism, memetic competition to find which concepts are useful and which aren't, and they will mutate accordingly. I don't know if in an Darwinian or Lamarckian sense, though. Also, what is a unicorn grounded to?

1.3 Formal Grammar

- Syntax vs. Semantix
- Semiotics
- We find that GPT also finds an inherent "semantic grammar" in the data. (but we do it differently)

undecidable problems:

- Determining if a context-free grammar generates all possible strings, or if it is ambiguous. Given two context-free grammars, determining whether they generate the same set of strings, or whether one generates a subset of the strings generated by the other, or whether there is any string at all that both generate.
- The problem of determining the Kolmogorov complexity of a string.
- Planning in a partially observable Markov decision process.
- Game of Life
- halting problem
- polymorphic typed lambda calculus (second-order lambda calculus)

1.4 Incompleteness Theorem

If our thoughts are indeed composed by some language, we can let our intuitions about its structure and limitations be guided by the advances made in the past century in regards to formal languages and computationalism. Gödel showed that any formal system strong enough to express Peano Arithmetic is incomplete [43]. Turing showed that any possible computation can be done by a Turing machine and concluded that the human mind must be computational [44]. [Piccinini disagrees]

1.5 Chomsky's Hierarchy

Chomsky's Hierarchy classifies formal languages of different strengths and the automata that recognize them [45]. Show the mapping of chomsky hierarchies onto the brain, relate it to [1] showing different DSLs in different brain regions.

Many contemporary versions of the LOT hypothesis (LOTH) argue that our conceptual framework is constructed bottom-up using some initial primitives. More so, the language we construct concepts in is executable, akin to a programming language [1]. The idea is that we model the world by creating programs that generate our observations [6].

Roumi et al.'s experiment indeed suggest that humans may interpret and compress sequence regularities in a type of program of minimum description length (MDL) [46]. Dehaene et al. further test this hypothesis and find similar results [1].

2 What is Truth?

- semantic closure

- David Deutsch: computation is a physical process, thought is a computation, therefore thought is limited by constructive mathematics/ computation, not by pure mathematics.

- constructivism, originality and plagiarism, everything comes from something, there are not isolated events

- computations are physical processes and are therefore restricted/ defined by physics and not by pure mathematics.

The question then remains, where primitives actually come from.

2.1 Where do the symbols/ primitives come from?

2.1.1 Neurosymbolic AI

[should this be in the concept section?]

Garcez et al. analyse connectivist and symbolic paradigms and find that the most effective AI would integrate these approaches [47]. They find that for effective learning from data, knowledge in neurosymbolic AI should be embedded into vector representations. Once networks are trained, symbols can be extracted. These symbols are not only descriptive but also operate at an optimal level of abstraction. This facilitates infinite applications with finite resources. Furthermore, this enables compositional and discrete reasoning at the symbolic layer, providing the capability to extrapolate beyond the given data distribution. Combinatorial reasoning is challenging. The combination of learning and reasoning addresses this problem by learning to reduce the number of effective combinations. Consider a neural network that learns certain regularities and recognizes that whenever it sees features of type A, features of type B are also present, but features of type C are not. This implicit rule can be made explicit by converting it into symbols: $\forall x A(x) \rightarrow (\exists y B(y) \wedge \neg \exists z C(z))$. [relate this to the framing problem that if A and B change, C should change...]

Discussing what LLMs are supposed to model, whether approximations are necessary etc. Symbols. [48]

2.1.2 Variable Binding

3 What is Meaning?

- what do the symbols mean? or "what's all that jazz (for concepts or so)" - is there a need for symbols?

Conceptual role semantics (CRS) is the view that the meanings of expressions of a language (or other symbol system) or the contents of mental states are determined or explained by the role of the expressions or mental states in thinking. The theory can be taken to be applicable to language in the ordinary sense, to mental representations, conceived of either as symbols in a 'language of thought' or as mental states such as beliefs, or to certain other sorts of symbol systems. CRS rejects the competing idea that thoughts have intrinsic content that is prior to the use of concepts in thought. According to CRS, meaning and content derive from use, not the other way round.

- variable binding
- grounding
- binding problem

Piantadosi asserts that symbols have no inherent meaning, instead, meaning emerges from the dynamic relations between symbols. I.e., only the conceptual role and the dynamics of the symbols define meaning [49].

By defining meaning as an emergent phenomenon of conceptual roles, Piantadosi and Hill do not rule out that large language models (LLM) already have some foundation of meaningful concepts [50].

Recent work tries to relate these ideas to biological systems, e.g. Quan et al. propose how role-filler interactions could be implemented in the brain via temporal synchrony to permit symbolic processing and dynamic binding of variables [51].

Santoro et al. argue for a semiotic approach, assert that symbols are subjective, and propose that meaning of symbols in artificial systems will arise when they are subjected to socio-cultural interactions [52].

Meaning: You establish the relationship between pattern and the function that describes the universe, your conceptual model, and that is meaning. Meaning is your entire mental universe. It's the unified model of reality that your mind is constructing. So for any new thing, if we can establish a relationship between the thing and our model, it has meaning. If we can place it in our conceptual framework, it gets meaning.

- subsubsection: what is meaning?

difference between what is meaning and what is meaningful, maps of meaning. what is meaningful is what guides you. its a gradient, a heuristic. Show studies that deeper understanding improves memory and meaning.

Ellis et al. use λ-calculus, which is Turing complete, to build something akin to a LOT [2]. In response, Piantadosi shows that combinatorial logic is equivalent to λ-calculus and why combinatorial-logic-like (LCL) language is preferable, for one because it doesn't assume primitives, thereby avoiding the problem of their meaning and origin.

One of the questions regarding a LOT is whether thoughts can be produced by simple syntactic symbol manipulation. I (and many others) suspect that semantics are not just an emergent phenomenon but that it actively shapes our perception and concept formation [52, 53].

Moreover, purely symbolic attempts, albeit being the original approach of artificial intelligence, have proven to be insufficient [47]. Instead, the successes of modern AI are owed to the advances of connectionist models. These however, have their own deficits, namely, being incredibly data-hungry, lacking causality, and lacking out-of-distribution generalization, to name a few. Therefore, new approaches try to combine these two AI strands into what is known as *neurosymbolic AI* [47].

3.0.1 Poverty of the Stimulus

should this be here?

The "poverty of the stimulus" argument is an argument that is often made in the field of linguistics and cognitive science. It argues that children are able to learn language despite being exposed to a relatively limited amount of data, which suggests that they must have some innate knowledge or ability that helps them learn language.

The argument is based on the observation that the linguistic input that children receive is often incomplete, ambiguous, or inconsistent. For example, children may hear sentences that contain errors or that are not grammatical. Despite this, children are able to learn the rules of their language and use them to produce and understand sentences that they have never heard before.

3.1 Abduction

- frame-problem
- IBE + abduction
- Markpoel 7 criteria
- Approximation?

Blokpoel et al. distinguish seven properties a computational model of abduction proper must conform to, namely isotropy, open-endedness, novelty, groundedness, sensibility, psychological realism, and computational tractability [54].

3.1.1 Abduction Proper

Abduction proper relates to the *generation* of hypotheses. In DreamCoder, this happens in the waking phase of the algorithm. When confronted with a task, DreamCoder comes up with the k best candidate hypotheses [1]. It does so by enumerative search under the recognition model Q. The neural network guides the search by pointing to the branches that are most likely to contain the programs to solve the task, thus narrowing the breadth of the search tree. In other words, branches of unpromising programs are being eliminated. By enumarative search using the recognition model, programs are chosen that are both the likeliest as well as the most succinct. The result is that programs are returned, enumerated by their posterior probability.

In the following I will go through the seven properties of abduction proper as outlined in Blokpoel et al. [54] and compare them to the implementation in DreamCoder.

3.1.1.1 Isotropy Fodor denotes that any knowledge that the system has, might be relevant to the solution of the task and should therefore be accessible [55]. DreamCoder does adhere to this principle, since all the knowledge it has is encoded in its library which is accessible in its entirety when generating hypotheses. Therefore, if a sub-program is relevant, it can be included in the proposed program.

3.1.1.2 Open-endedness The set of candidate hypotheses should be open-ended, meaning that it can contain any hypotheses which one could in principle infer. Here too, DreamCoder abides by the principle. All the programs that solve the task x could in principle be included by the set of candidate programs. The authors set the number of candidate hypotheses to $k = 5$ for efficiency. However, one could also list all candidate hypotheses without restrictions.

3.1.1.3 Novelty An agent should be able to generate novel hypotheses, i.e. programs that are not already stored in the library. In DreamCoder, programs are represented as polymorphically typed λ-expressions which allow for the ability to define new functions. When presented with a (novel) task, DreamCoder synthesizes programs, tailored to the task. This is one of the main features of the algorithm: program synthesis.

3.1.1.4 Groundedness A candidate hypothesis should be grounded, meaning, the relation of the representations of the explanandum and explanans should be well-defined. Since DreamCoder uses a DSL, this property is inherent in the system. Moreover, the output of DreamCoder is a library of programs (along with a recognition model). Programs are universal and can be represented by any Turing-complete language.

3.1.1.5 Sensibility A candidate hypothesis should explain the observation. All candidate programs suggested during the wake phase are sensible, as they are only selected during search if they solve the task at hand.

[1] Note that I will use hypothesis and program interchangeably, as when speaking about an hypothesis we mean a possible explanation to a problem, which in this case takes the form of programs.

3.1.1.6 Psychological realism

The computational mechanism which supports abduction proper should be biologically plausible. This property however, matters only if human abduction proper is to be replicated or studied. To study the mechanism itself this property is not a necessity. In any case, the wake-sleep algorithms is heavily inspired by our own sleep cycles and Lewis et al. show a remarkably similar finding in the human brain, interestingly using the same example of the analogy between an atom and a solar system used in Blokpoel et al., and previously Gentner in her original Structure-mapping Theory [56, 54, 57].

3.1.1.7 Computational tractability

Search is done by a specifically engineered parallel enumaration algorithm which combines Best-first and Depth-first search which run with $\mathcal{O}(n)$ and $\mathcal{O}(\log n)$, respectively. Hence, given the library, as well as the trained recognition model, abduction proper actually becomes tractable.

3.1.2 Inference to the Best Explanation

Now that we already have a set of candidates, how do we select the best of the generated hypotheses? After applying neurally guided search, we have an enumerated list of programs solving the task. *Best* is defined in DreamCoder as the shortest program with the highest probability of solving the task. This makes it seem as if abduction is solved easily and tractably. However, the bulk of the computations happen in the sleep phases. In the dreaming phase, the recognition model Q is trained to solve MAP inference, on both replays and fantasies and it is known for MAP inference to be NP-hard. Nonetheless, in replays the system only marginalizes over the best k programs for each task, which reduces the amount of computations and makes the model scalable. One of the major hinderences in the quest for commonsense in AI is the problem of finding the *relevant* pieces of knowledge while ignoring irrelevant information [2]. There are two main approaches trying to solve this problem. One is to improve the representation of the knowledge while the other posits a better inference should be applied. DreamCoder implements both these strategies; in fact, these two approaches bootstrap each other and keep improving while learning new tasks.

3.1.2.1 Representation

Regarding the representational approach, the question arises of how implicit knowledge should be dealt with. Say, $A > B$ and $B > C$, implicitly we know, $A > C$. This piece of information though, has to be derived before it can be used. Similarly, once this has been derived and has been made explicit, if any of the relations change, it might happen that a derived conclusion has to be adjusted retrospectively. This opens up new problems. One way of solving this is by using intrinsic representations which means that the representations of the relations have the same constraints as the actual relations [59]. In DreamCoder knowledge is represented symbolically, i.e. the programs in the library are implemented as syntax trees. During the abstraction phase, programs that have been found during wake are refactored. The authors leverage *E-graph matching* which is a data structure that can compactly represent alternative versions of the same program and *version space* which is a data-structure that allows for efficient set operations on its compactly represented programs, to deal with the combinatorial explosion that arises with refactoring. In essence, during abstraction the system looks for sub-routines that have been useful across many of the existing programs as well as the programs that have been newly found during wake and compresses them into new concepts which can be used in upcoming

[2] For an extensive discussion on the frame problem, see [58]

tasks. In this way, explicit knowledge is made more accessible and although the overall library grows, the depth and breadth of the library virtually shrinks because many sub-programs that are deep in the library don't have to be used anymore. Moreover, crucial implicit knowledge is therefore inferred and made explicit. Since the system develops and learns over time, it can adapt whenever it encounters novel tasks.

3.1.2.2 Inference Proponents of the inferential approach usually don't dismiss the symbolic representations but propose that developments in order to better understand inference itself must be made. This is exactly where IBE comes into play. During dreaming, DreamCoder strengthens the mapping between programs and tasks during *replay* and during *fantasies* it learns to deviate from the observed data-set. This means, that the system learns to quickly recognize which patterns in the library are useful and how to quickly find them. One could say, it learns an intuition for the task-domain. It can therefore extrapolate to new tasks.

During waking then, DreamCoder has an improved library of programs, as well as a better recognition model to navigate it. Crucially, this dual bootstrap between the two modes is what catapults the learning of the domain. An improved library with better abstractions and concepts leads to a better and more efficient usage of it which leads to better programs which can be found which in turn can be abstracted further.

- induction, deduction, abduction, inference to the best explanation
- "Although it is not clearly understood by many, classic grounded theory utilizes deduction, induction, and abduction as the necessary logic functions of the research process. Glaser's described the forms of logic—induction, abduction, and deduction—but referred to them as conceptualization, theoretical coding, and theoretical sampling. Induction begins with data and produces concepts, which are the building blocks of grounded theory. Employing abduction, the analyst infers relationships among the concepts to develop interrelated hypotheses. Deduction is used to gather data to fill in the gaps and produce an explanatory theory. Each type of logic is indispensable to classic grounded theory method. The purpose of this methodological paper is to briefly describe the process and product of each type of logic as applied to the language and procedures of classic grounded theory. - The Logic and Language of Classic Grounded Theory: Induction, Abduction, and Deduction" [is this related to the way concepts share information and are constructed? top down (deduction), bottom up (induction), lateral (abduction), maybe not because different truths are established? check this]

3.2 Subsymbolic Parse Trees

Implicit vs explicit

- GLOM also related to NCA and to holarchies
- Kissner
- JEPA
- etc.
- other world models, schidthuber, etc.

4 What Is a Computation?

4.1 Computational Mind

- RNA computation [paper]
- LCL, etc.
- Church-Turing Thesis
- Difference in computation between brain (Piccinini), ANNs and symbolic processing
- Is computation the only way to truth (relate this to constructivism)
- computational irreducibility: we can use generative models as a shortcut sometimes. we can compress regularities. But Wolfram shows that sometimes that is impossible. if truth is whatever is computable, and we take note of the idea of computational irreducibility, it means that there are some computations in which we must apply every step to get there. there is no shortcut.
- Do we have one generative model or multiple competing models (multiple DSLs, trying to find the best one)? Or do we just have competing representations at the frontier and it is more difficult to change beliefs deeper in the conceptual space, where concepts are foundational and others rely on?
- Jerome Busemeyer uses Quantum formulations to show how we are in a super-position of beliefs until we collapse them upon evidence.

5 What is a thought?

- Explain the idea of a thought as a trajectory through semantic space.
- How does this trajectory look like in LLMs? (Wolfram)
- How does this trajectory look like in GFlowNet? (perhaps operationalised as a Markovian trajectory?)
- How does a thought look like in enactivist/ purely reactive or reflexive cognitive frameworks.
- Thought as constructing probabilistic programs
- What does it mean to have a model?
- What do we model? Res extensa, Res cogitans. The inside as well as the outside world (Reference Solms, JB, Decartes)
- What does it mean to compute?
- Can any concept be definitively defined as a particular concept (either symbolic or symbolic vector e.g. SPA, hyperdimensional vectors) or are all concepts approximations?

6 What Does It Mean to Understand?

6.0.1 Operationalising "Understanding"

- Reverse Engineering: Is understanding being able to reconstruct the thing we want to understand? If you can break down an object or concept and then reconstruct it, perhaps it implies you understand it.
- Modelling: Understanding might be proportional to the model we create of a concept. The more accurate and comprehensive our model, the greater our understanding.
- Utilization: It could also be the degree to which we can use a concept. If we can apply it effectively, we could claim to understand it.
- Maybe understanding is the size/ strength of a concept? Imagine it as blobs in an abstract space, a large blob with many connections would be understood, whereas a new little blob with little connections is not.

What does it mean to truly understand something?

I would say that understanding requires causality, not just correlation. (Searle's chinese room)

Is it possible for artificial intelligence to grasp the essence of a concept like "body" or "mass"? It can approximate to an arbitrary degree but you can't square the circle, i.e. approximations at the limit do not become the thing approximated, they are just treated as such. This is related to joscha bach's interpretation of Gödel's theorem, that it cannot be implemented since there are no infinities in physical reality. Anyways, approximations might be enough, and might be what we're doing too. This is important for the symbolic vs connectionist debate.

GPT learns through the context in which words and phrases appear, essentially creating a concept network of words. The AI analyses this abstract relational structure to generate responses. Yet, we might question whether this process equates to understanding.

The conundrum is reminiscent of the mary's room thought experiment, in which Mary, despite having all the data about colour, has never truly experienced it. If she were to experience colour one day, would she glean any new information that was not already available in the data she had studied?

The question seems to probe whether phenomenological experience can be simulated through abstract data. While it's possible to argue that a blind man will never truly see just by reading about it (in braille), we must also consider that our experiences are essentially the result of neurons firing in our brains. If we possess the necessary cognitive mechanisms, it's plausible that phenomenological experience could be replicated by alternative means.

7 What is a Concept?

7.1 Identity and Essence

How do we discern between things?

Here we discuss homeostasis, allostasis — me against the world

7.1.1 Categories

- Prototype Theory (+ prototype, stereotype, archetype) (types and relation to programming?)
- intension extension?
- In LLMs concepts are defined by the way they are used. By correlation. By function. It could be defined by its properties, by its causal function, etc.
- LLMs
- mereology
- ontology
- substance theory
- process theory
- krakauer information individuality
- Relationship between agent and environment. Embeddedness, we are inseperable; Üexküll called this bio-semiotics and the extended phenotype (the spiderweb is part of the spider, humans domesticated fruit, animals, etc. these are all part of what it means to be human.)

7.1.2 Computational models of categorization

In their study, Zeithamova et al. delve into two predominant models of concept representations: and . The former views categories as generalized representations where category membership hinges on an object's similarity to a prototype, while the latter represents categories through specific exemplars, basing membership on an object's similarity to these stored exemplars [60]. Their findings suggest that during concept learning and category processing, high-level abstractions, such as associating an airplane with a car due to their shared role in transportation, are processed in the dorso-lateral pre-frontal cortex, associated with β oscillations and bottom-up processing. On the other hand, low-level abstractions, like identifying a new cat based on prior knowledge, are linked to the ventro-lateral pre-frontal cortex, involving γ oscillations and top-down processing—this latter approach resonating more with object-recognition or pattern-matching challenges.

relation to FEP, generative model, sys1,2 etc.

Feature-based models: These models represent categories as a set of defining features. They assume that category membership is based on the presence or absence of specific features.

Connectionist models: These models represent categories as patterns of activation in a neural network. They assume that category membership is based on the pattern of activation that results from processing sensory input.

Bayesian models: These models represent categories as a probability distribution over features. They assume that category membership is based on the likelihood that an object belongs to a particular category given its features.

- I think categories, similar to any self organizing system maintain a boundary of what they are and what they are not. they themselves can be thought as agents. We have the concepts but our perceptions are shaped, mostly by evolution and society. Our subjective interpretation is actually

minimal. Our perception of objects strengthens in concordance with the crowd, the society. So these concepts, which are sort of memes can be seen as agents themselves. They self-organise, and have a mapping to the actual underlying brain structure which is a mapping of the environment, to solve certain problems. a concept is the minimization or optimization of free variables in an active inference frame. This relates to gibsons affordances. so concepts might be formed also by function, or the relation of how i could interact with the concept. but concepts dont always have to be functional. this is also related to meaning. the function of a concept is its meaning. in that way it is the set of inferences one could make from said concept. of course it doesnt mean that it is meaningful, which is subjective to the observer and dependent on personal preference and goals. concepts are agents that serve a goal. notice how concepts have different levels of granularity, according to our interaction with them.

7.2 concepts

- is a concept a symbol?
- Do concepts exist?
- JBs interpretation
- Semiotics
- Symbolic vs distributed representation of symbols
- Are concepts discrete or continuous (symbol vs connectionsim)
- Do concepts exist?
- Is the self a concept?
- the transcendentals as the axioms of ontology (what are the eigenvectors of cognition?)
- proto-concepts
- concepts vs no concepts
- concepts as pointers
- normalizing over sensory data -¿ features -¿ objects -¿ concepts
- Concepts could be "chunked" into hyperdimensional vectors

7.3 How do concepts form

- Concepts are the address space of mental representations. They are merely pointers. - There's a hierarchy (holarchy) of abstraction of mental representations - The least abstract thing are impulses that come from sensory neurons. - Basically, discernible difference, which is how we define information, is the closest to physical reality. - We organize them into features. features describe how reality changes in the aggregate from moment to moment. - Features are arranged into objects - Objects remain stable if the perspective changes - Concepts abstract over all objects that belong to a group - the extension of a category - These concepts are organized in an embedding space

Conceptual space (relate this to Gordenford)

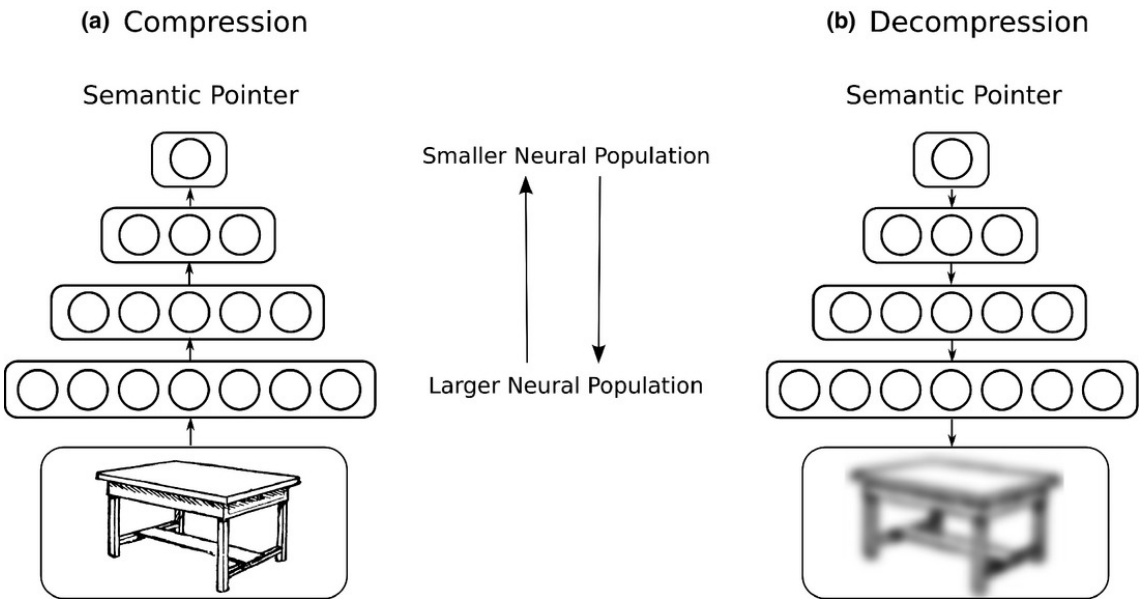

Figure 1.1: (a) A sensorimotor input, here a table, is compressed sequentially into a Semantic Pointer. (b) A Semantic Pointer is decompressed and returns a representation of a table. Due to the compression process the decompression results in noise. The figure is taken from Blouw et al. [3].

7.4 Semantic Pointer Architecture

Eliasmith et al. propose Semantic Pointers (SP) as neural representations that carry semantic content and can be thought of as vectors in high-dimensional space. They are composable into the representational structures necessary to support complex cognition [61]. To illustrate the notion of SPs, see figure 1.1. On the left we see how a visual percept, in this case a table, is compressed sequentially, much like in the layers of the visual cortex to eventually be represented by a single vector - a semantic pointer. On the right we see the "inverse" function in which a semantic pointer is decompressed and returns the table. Notice that due to noise, the result does not correspond exactly to the original input.

7.5 Concept space

Properties of concepts, introducing the idea of concepetual spaces and a geometry of concept space (latent space)

Gärdenfors offers a geometric interpretation of cognition that serves as a bridge between the sub-symbolic and the symbolic, between the perceptual and the conceptual [62]. Her argues that concepts can be represented as regions in a conceptual space, where the dimensions of the space correspond to the different features or properties that can be used to describe the concepts.

In this model, each quality dimension — be it temperature, weight, brightness, or hue — becomes an axis in a multi-dimensional space. Concepts are then regions within these spaces. Imagine, if you will, a three-dimensional space where color is represented: one axis for hue, another for saturation, and a third for brightness. A "concept" like 'red' would be a specific region in this 3D color space.

In mathematical terms, a set is convex if, for every pair of points within the set, the line segment connecting them is also contained within the set. Imagine a balloon. If you pick any two points

within the balloon and draw a line between them, that line never leaves the space of the balloon. In Gärdenfors' geometric concept space, most concepts are "convex" regions. For example, if you have a swan and a goose and consider them both as instances of "bird", then it's likely you'll consider everything in between also a "bird".

[expound on that, and relate to rest]

Gardenfords book (2004)

I suspect that we use symmetry as an inductive bias [relate to dehaene]. The concept of opposites is a type of symmetrical relation where two entities are diametrically opposed on a spectrum or scale. Day and night, hot and cold, up and down — these pairs may help us organize our world into understandable categories. Of course there are many other types of symmetry, which could act as important inductive biases in order to construct a conceptual world model,

Reflectional Symmetry When one half is the mirror image of the other half. E.g., many human faces or butterfly wings.

Rotational Symmetry When an object looks the same after a certain amount of rotation. E.g., a regular pentagon or a snowflake.

Translational Symmetry When an object can be moved (or translated) a certain distance and appear unchanged. E.g., wallpaper patterns.

Bilateral Symmetry Seen in many animals, where the left and right halves of an organism are mirror images of each other.

Radial Symmetry Where symmetry is centered around a central axis, like in starfish or daisies.

Spiral Symmetry Seen in objects like shells or certain galaxies, where there's a continuous curve centered on a point.

Chiral Symmetry Refers to objects that are mirror images but not superimposable, like left and right hands.

Time Symmetry In physics, certain processes are time-symmetric, meaning they can proceed forward or backward in time without violating the laws of physics.

Scale Symmetry When patterns look the same at any scale. Fractals are an example of this.

Oppositional Symmetry In abstract terms, where concepts have opposites or counterparts, like good/evil, light/dark, or positive/negative.

isn't that already inherent since ANNs are doing linear algebra? I think the ANNs are capable of finding symmetrical relations but it could be a more explicit inductive bias/ heuristic? like we see that LLMs find some relations (man : king :: woman : queen), but it could be used explicitly for the construction of concepts. Also think about relational

[insert figure]

7.6 Hemispheric Lateralisation

- notion that if we don't have a concept for it, it does not exist for us. I.e. even if our sensory equipment can in principle take it in, and does take it in, it is not there for us. It seems we

are unable to attend to it. Perhaps attribute meaning to it and therefore it doesn't enter our conscious awareness.

- refer to Wittgenstein, Alfred Korbinszky, etc. analogy with chess

argument for hemispheric lateralization

- Broad vs narrow attention. One sees the trees, the other the forest.
- the left wants to jump to conclusions and is much more quick and dirty, while the right says hang on
- left deals with things that are known, when something isn't, its better to leave it to the right until it can be categorised by the left
- things in the left are more isolated, while in the right everything is connected to everything else
- left is static, right is flowing and changing
- the left abstracts, the left is more interested in categories, the right in the unique case
- the left sees things as inanimate, the right sees them as animate
- as we age we use the representations of what we know increasingly, neglecting the world. We become more solipsistic. This makes sense because perceiving things for the first time is computationally heavy and expensive, so the things we know from experience will usually be good enough.
- Talk about confabulation (and perhaps in relation to LLMs)

Often, the most obvious things, once unpacked, reveal the most [examples?]. Our most basic perceptions and assumptions are so deeply integrated into our cognitive processes that they become invisible to us, essentially automatic. When driving the same route everyday, people may arrive at their destination without conscious awareness of it. Tasks become automated. [show studies, other examples like piano.]

From a computational standpoint, one could say that these deeply embedded perceptions function much like 'compiled code' in a computer program — efficient and fast, but not easily inspected or altered. These perceptions are model-based representations optimized for computational efficiency, trading off flexibility for speed. They exist as pre-computed "shortcuts" that enable us to interact with the world without incurring high computational costs each time we encounter a familiar situation.

In machine learning, this relates to the trade-off between model-based and model-free learning. Model-free approaches require higher computational costs upfront but are more adaptable. Model-based strategies offer efficient but rigid ways to interact with the world. Both strategies have their pros and cons, reflecting a trade-off between computational efficiency and cognitive flexibility. [source]

As we age, we rely increasingly on these internal representations instead of engaging directly with external reality, making our view of the world more solipsistic. [source]

think of two types of intelligences (crystal vs fluid ?), also exploration exploitation

We can also include barbara oakley on types of focus, deep and direct, vs broad

This phenomenon can be explained by the increasing reliance on "cached" or "memoized" cognitive models, which offer a computationally cheaper way to navigate reality. It's a form of cognitive "greedy algorithm," opting for local optimizations based on past experience rather than recalculating the optimal path each time. [source + relation to predictive coding paradigm]

This concept parallels the notion of "overfitting" in machine learning, where a model becomes so tailored to the training data that it loses generalizability. In human cognition, an over-reliance on established mental models could result in decreased adaptability and an insular worldview, effectively isolating the individual from the ever-changing external environment.

As we automate more cognitive functions—either through learned habits or technological aids—we engage less in "active sampling" of the world, reducing the richness and nuance in our perceptions. This leads to a form of "lossy compression" of reality, where only the most salient features, according to our models, are retained.

- Problems should be solved at the lowest possible level of the hierarchy. Larger goals require more sophisticated organization. - How does a system recognize that it cannot solve it alone and that it needs to recruit other cells? Think of the computational boundary. Perhaps we must be able to estimate computational complexity. - This compression of sensory information into concepts may be the origin of symbolic processing.

- Things we don't have concepts for don't exist. We can't describe them in our language. [related to joscha about truth.]

- show representations in human brains and argue why the brain may largely be a generative model.

A paradigm, coming originally from Ethology is the idea of reactive agents which have four defined "rules" on which they act (Murphy, 2000). Reflexes are hard-wired responses to certain stimuli. A knee-jerk reaction. Taxes define the movement of an organism towards (or away from) a certain stimuli. An example is the way ants follow a trail by chemotaxis. More complex behaviour is elicited through fixed-action-patterns. These consist of a sequence of certain behaviours, usually triggered by some stimulus. A squirrel's nut burying behaviour is an example. Lastly, the sequencing of innate behaviours such as a digger wasp's brood-tending is an example of even more complex behaviour. Further, it has been shown how complex behaviour can emerge as an epiphenomenon such as in swarms, ants (Franks, 1989) or even in Conway's popular Game of Life (Conway, 1970).

- Efficient inference over hierarchical Bayesian models is still difficult.
- Approximate inference
- Here we can go into models monte carlo e.g. etc.
- Einstein: intuition, rationality

8 Cognition, Nested Multi-scale Hierarchies, and Self-Organisation

Biological systems demonstrate remarkable complexity through self-organization, a process where spontaneous pattern formation results from the interactions of individual components within an environment.

Biological organization is composed of nested goal-seeking agents, maintaining operational and organizational closure through homeostasis and allostasis [63, 64] [explain terms in more detail, or is glossary enough?].

Atoms form molecules, which form cells, tissues, organs, organisms, groups, societies, civilizations, and so on. Levin describes this as a multiscale competency architecture, which is not only structural but also functional [65]. Each level of this organization has some competency and solves problems in its own action space. Moreover, each level interacts with the levels above and below which is reflected in the idea of circular causality [63, 64].

9 Who am I?

In order to understand who I am, I need to understand what I am not. What is a cognitive system, how does it emerge and how does it develop. How does a cognitive system make the distinction between it and the world?

9.1 The Genesis of Cognition

When thinking about the genesis of the self, we can think back to our earliest developmental stage, that of being an unfertilised , and realize that the self gradually emerges from it.

Levin shows that upon scratching a , it self-organizes into twins or triplets, etc., depending on the amount of scratches.

source

. This demonstrates that the number of selves in an is not determined by genetics, but is instead a result of physiological self-organization. The problem of discerning between one's self and the external world is an ongoing, dynamic process that needs to be solved "online". This indicates that our self-concept is ever-evolving, reshaped by new information and experiences.

The ability to pursue goals is a trait that all agents have in common and the size of that goal is a way to classify and compare diverse intelligences, Levin suggests [66, 67].

For example, a single-celled organism may have the goal of finding food and avoiding predators, while a human may have the goal of building a family or writing a thesis about cognition.

is the homunculus is basically an advanced version of this?

The brain builds a map of proprioception. It's a statistical map over nerves that fire together. When they almost always fire together, they're likely to be close to each other. Homunculus.

Now the way we distinguish between us and other people or more generally, us and the rest of the world, is simply by realizing that my actions lead to changes in my proprioception and other's actions do not. I.e. we perceive a change in the environment and we need to classify whether we produced the change or not. If the change and our sense of self aligns, it's likely we did it. [elaborate]

In

source

Levin expounds on the idea that the self-organisational process of bodies is fundamentally the same as that of minds.

not only in structure but in representation. i.e. can we show that the imagination, the conceptual world has the same structure as the actual hardware? is the software built by the same principle as the hardware? is it growing simultaneosly? is there an isomorphism?

.

All bodies are *composed* of a multitude of levels of organisation, from single cells to entire organisms. Each tier of this hierarchy possesses unique competencies and goals, yet collaboratively works towards the organism's overarching objectives.

is the grounding of cognitive processes in bodily experiences. Bodies are not just passive vessels for minds but play an active role in cognition

Serge paper on embodiment, sensorimotor information

[elaborate].

Diving deeper into bodily self-organization, organisms such as axolotls exhibit a fascinating ability. Despite undergoing amputation, they can regenerate their limbs and halt precisely when restoration is complete. This ability, despite the variety of starting conditions and challenges, hints at an inherent bodily blueprint or internal representation. Intriguingly, this map is underpinned by bioelectrical patterns, which play a pivotal role not just in regeneration but also embryonic development.

Expanding on this, Levin notes that there is no marked distinction between neural and developmental tissue in this context. Both comprise ion channels and electrical synapses. In fact, novel techniques have emerged to both interpret and modify these bioelectrical blueprints.

source

These "blueprints", or more precisely, endogenous bioelectric pre-patterns, guide the cells to arrange in a certain way and by manipulating them, for instance by introducing potassium ion channel RNA, Levin and his team successfully induced cells to form eyes in non-ocular locations. More impressively, in the absence of sufficient cells, the system recruits additional cells to realize this goal.

In one experiment, the team modified the bioelectrical pattern of a planarian to have a two-headed form. Upon injury, the planarian regenerated, aligning with this new, dual-headed pattern. This reveals, that the planarian is able to have a representation of a body state, different from its current state. It means it has a rewritable memory and is able to represent .

This finding suggests, congruent to the conventional view, that genetic information alone doesn't solely dictate the form and function of organisms. Instead, bioelectric signals play a pivotal role in guiding cellular behaviors and tissue outcomes. Traditional genetics emphasizes the idea that the DNA sequence encodes the necessary information to produce an organism's phenotype. While this is fundamentally true, there seem to be other layers of information that can guide tissue organization and even regeneration.

This may be a primitive precursor to symbolic thought. Perhaps even to visualization and imagination. Perhaps moving from bioelectrical pattern representation of the brain, which is in a sense the first counterfactual, this may be the first primitive version of symbolic? representation, and allude to the brain being a machine to store more complex patterns.

Show the work of Lenia, how they do it, and what they have achieved. Also, the Gecko thingy from Distill.pub

9.2 Computational boundary of the self

Levin introduces the term cognitive light cone, which metaphorically describes the limits of an agent's knowledge and ability to act. It is defined by the set of all events that an agent can perceive, measure, and model.

introduce cognitive light cone and include stuff from com-

9.2.1 Body Patterning and Cognition: A Common Origin

Cells are able to make group decisions and regrow into defined patterns.

There is a strong metaphor between "adaptive whole organism behavior and the plasticity of cellular activity during construction and repair of a body"

"Neural networks control the movement of a body in three-dimensional space; this scheme may be an evolutionary exaptation and speed-optimization of a more ancient, slower role of bioelectrical signaling: the movement of body configuration through anatomical morphospace during embryogenesis, repair, and remodeling"

9.2.2 Multicellularity vs. Cancer: The Shifting Boundary of the Self

While healthy cells within a multicellular organism orchestrate their functions to achieve systemic homeostasis, cancerous cells behave in a manner that one might describe as "narcissistically individualistic," focused solely on their own objectives.

In standard physiological states, cellular networks coalesce to fulfill an overarching agenda—namely, the creation and sustenance of intricate macrostructures. Here, individual cells are subsumed into a collective intelligence, a sort of physiological "hive mind" or "swarm intelligence", directed toward system-level anatomical objectives. Contrarily, in the cancerous state, a cell's conception of its 'Self' is shrunk, both spatially and temporally.

Spatially, cancerous cells find themselves in a state of electrical isolation from their cellular neighbors, impairing their capacity for long-distance information exchange and cooperative action. Temporally, the scope of a cell's prospective planning dwindles: where a healthy cell may operate on a time scale spanning decades, a cancerous cell may focus solely on activities that, though enhancing its own immediate survival, jeopardize the host organism in the near term.

Unicellular organisms or cancer cells are not intrinsically more selfish than cells in a multicellular organism; rather, the "Self" is defined as the boundary of information, in terms of space, time, and complexity, that can pass between the subunits.

Most biological organisms emerges as a hierarchy of nested "selves," each subject to its own evolutionary pressures and game-theoretic dynamics, at varying scales of organizational complexity.

This conceptualization of a dynamic boundary of the 'Self' has implications that stretch beyond the biological realm, into the study of swarm intelligence. Consider, for instance, a termite colony as a "super-organism," with its own extended physiological and rudimentary cognitive functions. This interplay between patterns of cellular organization and larger sociobiological systems offers a poignant example of the scale-invariance that characterizes cognitive paradigms, equally applicable to cellular communities and animal (and human) societies.

9.2.3 Defining Individuation From a Cognitive Perspective

Levin proposes a definition of an individual based on its information-processing structure and the goals it pursues. A coherent, unified self emerges from the integrated activity of its components that work towards specific goal states. This definition can apply to both biological and artificial agents, focusing on the information processing and goal-directed activity of any given system.

The cognitive boundary of an individual is the most distant set of events that a system can measure and attempts to regulate in its goal-directed activity. Different agents have different cognitive boundaries based on their complexity and abilities. For example, a tick has a smaller cognitive boundary than a dog or a human due to its limited memory and predictive power.

Individuals can exist at different levels of organization, with various, sometimes overlapping goals. This framework can be applied to ethology, artificial intelligence (AI), and artificial life. However, more work is needed to make it practical and applicable in these fields.

The volume of possible cognitive boundaries expands over evolutionary time. Innovations in body structure can drastically increase the cognitive boundaries of viable selves. There may be no upper limit to the size of cognitive boundaries, and it's possible that artificial life, biomedical enhancement, or engineered AI could create beings with much larger cognitive boundaries than humans.

Levin concludes that a potential sequence of evolutionary scenarios for the expansion of cognitive boundaries. "Thus, physiological connectivity is the binding mechanism responsible for the appearance of larger unified Selves."

The expansion of cognitive boundaries in the biosphere can be traced through a hypothesized sequence of phases, beginning with homeostasis. Homeostatic persistence allows simple systems to maintain spatial and metabolic integrity, providing the basis for cognitive goals. The minimization of anti-homeostatic stress is a powerful driver for evolutionary innovation, leading to the development of richer biochemical networks and memory systems.

One example of an early memory system is chemotaxis in bacteria. More advanced creatures developed complex learning networks, allowing for associative learning and modularity, which in turn led to the evolution of reactive homeostasis into predictive allostasis.

Sensory machinery in complex animals is based on ancient ion channel mechanisms. As cells organize into multicellular tissues, they can share information and expand their cognitive boundaries. The advent of multicellularity results from the drive to minimize surprise, with multicellular organisms providing more predictable environments for individual cells. This leads to the formation of more complex agents with larger goals.

The transition from single cells to multicellular tissues is facilitated by developmental bioelectricity, which enables the exchange of ionic signals among cells to propagate instructive influence. This has been observed in bacterial biofilms and more advanced cell types. Gap junctions, or intracellular electrical synapses, allow cells to selectively share bioelectric states and implement memory. Physiological connectivity creates larger unified Selves, with bioelectric networks responsible for coordinating cells toward common goals, such as body patterning.

The developmental control bioelectric network shares an evolutionary history with the nervous system, and neurotransmitter molecules are used downstream of bioelectric driving forces. Tunneling nanotubes, axon-like structures with gap junctions at their ends, enable directed connections between cells and pave the way for neural axons and electrical synapses. The bioelectric system was optimized for speed in nervous systems, using the same mechanisms to optimize input-output relations within and between internal mechanisms and the outside world.

In conclusion, the expansion of cognitive boundaries in the biosphere can be traced through various evolutionary stages, from homeostasis to multicellularity, and the development of complex learning networks and memory systems. This expansion is facilitated by the minimization of anti-homeostatic stress and the drive to minimize surprise, leading to the emergence of more complex agents with larger goals.

9.3 Embodied cognition and circular causality: on the role of constitutive autonomy in the reciprocal coupling of perception and action

[64] Varela defines cognition as *effective action*: action that preserves the agent's autonomy, maintaining the agent and its ontogeny, i.e., its continued development.

There are two hallmarks of a cognitive agent: 1. Prospection, i.e., prediction or anticipation 2. The ability to learn new knowledge by making sense of its interactions with the world around it and, in the process, enlarging its repertoire of effective actions.

Homeostasis refers to the maintenance of stable internal conditions within an organism, while allostasis refers to the adaptive regulation of these internal conditions in response to changing demands.

Both homeostasis and allostasis play a role in the reciprocal coupling of perception and action, as well as in the self-regulation of the agent's autonomy. (Perception-Action Cycle)

Autonomy is defined as the degree of self-determination of a system, i.e., the degree to which a system's behavior is not determined by the environment and, thus, the degree to which a system determines its own goals.

I would argue that the autonomy always depends on the levels above and below, in the hierarchy. i.e., we are part of a society, but we are also constituted of and encapsulated by simpler elements. Therefore we share the goals of those levels. Being part of an environment imposes an inductive bias.

Living systems face two challenges: they are

1. delicate: they are easily disrupted or destroyed by environmental forces, requiring avoidance or repair of disruptions. 2. dissipative: they are comprised of far-from-equilibrium processes, necessitating external energy sources to maintain their stability and avoid lapsing into thermodynamic equilibrium, which would threaten their autonomy or existence.

relate (or reference) this to maxwells explanation of active inference.

Cognition, is the property which helps look into the future, to avoid these problems.

There are two types of autonomy:

1. *Behavioral autonomy* focusses on the external characteristics of the system: the extent to which the agent sets its own goals and its robustness and flexibility in dealing with an uncertain and possibly precarious environment. 2. *Constitutive autonomy* focusses on the internal organization and the organizational processes that keep the system viable and maintain itself as an identifiable autonomous entity.

These two parts of autonomy enable each other recursively.

this is again FEP, homeostasis, allostasis

Operational closure refers to a system that is self-contained and parametrically coupled with its environment but not controlled by the environment. It is a characteristic of a system that maintains its autonomy by subordinating all changes to the maintenance of its organization. This concept is used to identify systems that are self-contained and exhibit self-production or self-construction.

Organizational closure, on the other hand, characterizes a system that exhibits self-production or self-construction and is recognized as a unity in the space where its processes exist. It is a necessary characteristic of autopoietic systems, which are self-organizing systems that self-produce and operate at the biochemical level. Organizational closure is related to the concept of constitutive autonomy, where a system actively generates and sustains its existence and systemic identity under

precarious conditions.

Both operational closure and organizational closure are important concepts in understanding the autonomy and self-regulation of cognitive agents. They highlight the self-contained nature of these systems and their ability to maintain their organization and identity through self-production and self-construction processes.

Structural coupling refers to mutual pertubation of an agent and its environment.

They conclude by highlighting the importance of constitutive autonomy and allostasis as a form of predictive regulation [relate this to FEP].

- Autopoiesis
- Heterarchy
- Cellular Automata
- Lenia
- (Pure) Enactivism?
- Show how the FEP (Bayesian interpretation) can be used to explain multiscale systems.

Difference between self-organization and emergence:

Self-organization is a process where a global pattern emerges from numerous lower-level component interactions, while emergence refers to the generation of a global pattern that is qualitatively different from the assembly of components.

Self-organization is basically a macro-level structure out of smaller parts, emergence is when the whole is more than the sum of its parts.

Self-organization in emergence leads to systems with clear identities or behaviors due to local-to-global determination and global-to-local determination.

relate this to piccininis distinctions of emergence. Also to michael levins observer dependent cognition (sth like that?) where function emerges by viewing the system in different ways.

Autonomous systems are not only reactive but also preemptive and proactive, readying themselves for multiple contingencies and deploying strategies to deal with them while pursuing their defined goals.

we are not only reactive

Such systems must therefore be able prepare for counterfactual states

link to counterfactuals

. Allostasis is concerned with adapting to change in order to achieve the goal of stability in the face of uncertain circumstances.

link to hemispheres. apprehension vs ?

Seth notes that "He notes that attention can then be viewed as a way of balancing active inference and model update, (referred to as precision weighting)."

10 What is a Holarchy?

A holon has an identity and that identity can be interpreted, which creates meaning.

The holon transcends and includes. It is dynamic and it changes. When it changes, it changes its identity and therefore, the meaning that can be derived from it.

An atom, when it accepts more protons and electrons, changes its identity and function and ability to connect with other atoms.

A holarchy is a system in which each whole is made up of parts, and each part is a whole in itself. Holarchies are nested, meaning that each whole is also a part of a larger whole. This structure is similar to a hierarchy, but with the important difference that holarchies are non-linear. In other words, there is no single top-down chain of command. Instead, each level of the holarchy is interconnected and interdependent.

The concept of holarchy was first introduced by Ken Wilber in the 1970s [source], an extrapolation of Arthur Koestler's notion of a Holon, a thing that is simultaneously a whole and part of a larger whole, following the cybernetic tradition of studying the complex dynamics of self-organizing systems. Wilber realized that the human mind and society could also be understood as holarchies, and he developed his integral theory to describe this structure in detail.

One of the key features of holarchies is that they are emergent. This means that the properties of a holarchy cannot be predicted by simply understanding the properties of its parts. For example, the behavior of a human brain cannot be fully explained by understanding the behavior of individual neurons. Instead, the brain's emergent properties, such as consciousness and intelligence, arise from the complex interactions of its many parts.

- argue thoroughly why the conceptual framework is built as a holarchy
- We agree on an assumption that we construct a model of the world.
- What exactly would an explicit model or an implicit model or no model at all look like?
- We agree that there are certain hierarchical relationships. [example book, music, ontology, etc.]
- Here we can talk about conceptual structures, holarchy, part-whole hierarchies.
- How concepts are split with more attention, etc.
- Do the upper levels of a holarchy control the lower levels? I.e. Does the holarchy become a heterarchy?
- Heterarchy, sand example in nested organism pregnancy friston paper, DNA, GEB
- Multi-scale systems
- Ontology
- Mereology? and relation to Category theory?
- This is essentially learning a
- hierarchical LVM
- hierarchical graphical model
- Parse tree
- Can be formalised as Context Free Grammar.

- The grammar here is the generative model.
- What is a generative model?
- A generative model is simply a probabilistic description of how causes (i.e., latent states) generate consequences (i.e., data or sensations). [8]
- Include Hofstader levels of abstraction (let's have a coffee.)

10.1 Active Inference

[12] Ramstead et al. make the distinction between *active* inference and *enactive* inference.

"active inference," is a corollary Karl Friston's Free Energy Principle [source]. This framework posits that living systems minimize a variational free energy function to maintain themselves in a state that is compatible with their existence. Here, cognition is essentially about predictive modeling, and action is how an agent resamples the environment to minimize the error in its predictions. The agent, in this frame, is akin to a scientist, continually updating its internal models of the world to reduce surprises and fulfill its own expectations. It's a constant renegotiation of hypothesis and test, theory and experiment, query and observation.

"Enactive inference," on the other hand, draws more from the "enactive" tradition of cognitive science, inspired by figures like Francisco Varela and Evan Thompson [source]. This perspective argues that cognition is fundamentally rooted in an organism's interactions with its environment, emphasizing a co-constitutive relationship between the two. There's a deeper focus here on the importance of embodiment and the constitutive role that action plays in shaping cognitive structures. Mind and world, subject and object, create each other in a reciprocal embrace. It's less about a model inside the head predicting the world and more about the continual creation of a world through interaction.

From Maxwell Ramstead Tutorial on active inference

Caucheteux et al. provide evidence that the human brain continuously predicts a hierarchy of representations that spans multiple timescales and also that syntactic and semantic predictions are processed separately [68]. This differs from LLMs which are trained to predict the next token. [maybe put this in language section]

The complexity of the human brain arises from its intricate structure, both in terms of its physical components and its functional organization. A comprehensive understanding of the brain requires a multilevel approach that spans from the microscopic scale of molecules to the macroscopic scale of brain systems and interactions with the environment.

At the most foundational level, the brain's function rests on the interactions of its cellular components. Neurons, the primary cellular units of the brain, are responsible for processing and transmitting information. These transmission processes are facilitated by neurotransmitters, the chemicals that help propagate signals across synapses, the specialized junctions between neurons.

Beyond individual neurons, the microcircuit level involves small neural networks. These are clusters of interconnected neurons that serve specific functions. Whether it's processing particular sensory information or executing a distinct motor task, these microcircuits play a pivotal role in the brain's operational hierarchy.

Scaling upwards, the brain is segmented into specific brain areas and nuclei, each playing a unique functional role. For instance, the hippocampus is crucial for memory processes, while the basal ganglia are deeply involved in movement and motor coordination.

At the systems level, individual brain areas are not isolated entities but are part of broader brain systems. These systems, such as the limbic system responsible for emotion and motivation, comprise various interrelated structures, e.g., the amygdala, hippocampus, and certain sections of the thalamus.

[is this section too long? maybe shorten it, we get it, the brain is a hierarchy..]

At the pinnacle of brain organization, we find the integration of all its parts. The brain's two hemispheres, the left and right, offer different functional specialties. For instance, in a majority of right-handed individuals, language processing predominantly occurs in the left hemisphere [go into McGilchrist hemisphere differences? or in sys1,2, section?]. Furthermore, the integration of the brain and spinal cord establishes the central nervous system, crucial for sensory information relay and motor command execution.

The brain does not function in isolation. It's in a continuous dialogue with the body and the external environment. The brain-body interaction underlines the significance of feedback loops that shape brain processes. Simultaneously, social and environmental interactions influence the brain, underscoring the interdependence between an individual and their surroundings.

[69] One can witness that the brain, in its form and function, encapsulates a hierarchy that reflects the multifaceted layers of its surroundings. A paper by Park & Friston goes further to demonstrate that the brain showcases nested networks that include structures as micro as dendrites to as macro as entire brain regions.

These multi-scale systems operate not only in the spatial dimension, from microcosm to macrocosm, but also temporally. They span across phylogenetic, epigenetic, and ontogenetic timescales. This poses a challenge: how can we address these diverse spatial and temporal scales in a principled manner? What framework can effectively encompass this vast range of scales?

Most natural systems that are self-organizing tend to dissipate, as highlighted by Vernon [reference]. They increase entropy, which in this context, can be perceived as a measure of potential configurations a system could inhabit. A system with high entropy has a myriad of configurations available, while one with low entropy has limited options.

Surprisingly, about 80% of the brain's connections are involved in feedback [source, predictive coding]. Such a significant portion of the brain's energy is invested in a system that seems to serve a feedback purpose.

The predictive processing paradigm posits that instead of viewing the brain as a passive data collector, the brain is a query mechanism, primarily engaged in top-down prediction. The primary function becomes producing predictions about imminent sensory inputs. What ascends is the prediction error, the mismatch between predictions and actual sensory input. In this framework, perception becomes an inquiry-response mechanism, much like a Google search. The brain, through active inference, seeks to reverse the causality arrow, shifting from received data to inferring the probable causal factors.

This continuous feedback mechanism constantly compares expectations with actual perceptions. To minimize discrepancies, the brain can either modify its model or alter its environment. How do we quantify the effectiveness of such a model?

A common approach involves constructing various models encoding diverse hypotheses and evaluating their ability to account for data variance. Condensing this process yields the concept of *variational free energy*. Crucially, it's not analogous to thermodynamic energy; it gauges how efficiently a model explains data variance.

The term 'variational free energy' draws parallels with thermodynamics. In thermodynamics,

free energy represents the residual energy in a system capable of work. In information theory, it denotes the remaining room for parameter adjustments.

The concept of the Markov blanket suggests that given a set of states, the sensory and active states of a system remain independent of external world states. Active inference narrates how internal (model-encoding) and active (akin to skeletal muscles) states adapt to minimize free energy, enabling inference.

Every component in this complex system is itself a system, from entire organisms to individual cells. Each component, being Markov-blanketed, can infer its position relative to others, provided communication exists. By sharing a generative model, units at one level can enact target morphologies.

10.2 Self-organization and predictive processing

[8] As the brain develops, its neurons self-organize into networks that are able to generate and test predictions about the world. This process of self-organization allows the brain to develop a model of the world that it can use to make inferences about sensory input.

One way to think about the relationship between self-organization and predictive processing is to consider how self-organization may allow the brain to develop a model of the world. As the brain develops, its neurons self-organize into networks that are able to represent different aspects of the world. For example, some networks may represent the visual world, while others may represent the auditory world or the somatosensory world.

These networks are constantly interacting with each other, and this interaction allows the brain to develop a unified model of the world. This model of the world can then be used to generate predictions about sensory input.

The identity of a multi-scale system can be thought of as the emergent properties that arise from the interactions of the different levels of organization. For example, the identity of a cell is not simply the sum of the identities of its individual molecules. Rather, the cell's identity is emergent from the way in which the molecules interact with each other to form complex structures and networks. [relate to levins part]

This is reminiscient of embedding spaces in which words are self-organizing by getting input from the world.

LLMs also create a generative model, but lack the causal structure.

can causal structure, understanding, etc. be approximated?

- Discernment of Concepts
- Discernment as the primary operator.
- Me against the World
- Homeostasis
- Allostasis
- Operational Closure
- Organizational Closure
- Intentional Stance

- Active inference
- Explain Bayes theorem and how it relates

Free will: if i can out model you, i.e. i can predict every action you do (we can try this with simple organisms.)

- Discuss Bayesian causal inference
- Amortized sampling/ amortized inference
- Are these the most promising tools to research complex systems?
- Do we need a language of thought? (reference to JB, a chair isnt true or false, only a description can be true of false)
- Then, does the language have to be symbolic, or not?
- Can a representation of a word, of natural language be seen as a symbol? yes, it points to something, and we know that we can work in language without the meaning, the underlying understanding.

11 Implications/ Consequences of the theory

- Memes as exlicit cultural concepts of a super-organism
- the conceptual world, just like the physical/ material world has a multi-scale structure. We can extend Dawkins idea of memes and realise that conglomerated ideas, form ideologies [what is the right scale?] cells -¿ organism -¿ etc. :: thought -¿ idea -¿ ideology, etc. [question] when do LLMs form beliefs, ideologies
- Cognitive light cone view of cancer and psychopaths (also in a game-theoretic view)
- Describing systems as levels in a holarchy. We can describe groups, families, companies, corporations, (think noah harari and money is invented) as an organism and ask whether it is conscious.
- What we perceive is the simulation of our generative model. [relation to baudrillard, simulation simulatcrum]
- Summarise main findings and especially relate them to the questions. What have we learned about identity and the question "Who am I?"
- Understanding, goals, models, etc.

11.1 review of human capabilities and comparing PPL vs LLMs

11.1.1 World Model

11.1.2 Bayesian Model

11.1.3 Compositionality

11.1.4 Causality

11.1.5 Correlation Does Not Imply Causation

Generative models assign probability distributions, representing observations in such a way that they can generate new data points similar to the observations. The mapping between the probability distributions and the data is correlational.

Causal models in contrast, represent hypotheses of how observations were generated. They do not just represent the observations by correlation, but aim to accurately reflect the mechanisms by which the data originated.[source]

[The Reichenbach Principle, named after philosopher Hans Reichenbach, is a foundational concept in the philosophy of science, particularly in the study of causation. It essentially states that if two events are found to be statistically correlated, then there must exist some causal connection between them. This causal connection can manifest in three forms: either one event causes the other, the second event causes the first, or both events are influenced by a common cause. This principle highlights the difference between mere correlation (statistical association) and causation (a direct or indirect relationship leading to the occurrence of an event).

The significance of the Reichenbach Principle becomes clear when considering why causal models contain more information than purely statistical ones. Statistical models excel at identifying and quantifying relationships between variables based on observed data. They can tell us, for example, that two variables tend to vary together in a predictable way. However, statistical models don't inherently provide information about the directionality or the underlying mechanism of these relationships – they don't tell us whether one variable causes changes in the other, or if both are influenced by an external factor.

Causal models, in contrast, are designed to map out these directional relationships. They go beyond mere association to explain how one variable affects another. This is done by incorporating knowledge of the underlying mechanisms and processes that drive the relationships between variables. In a causal model, variables are not just connected; they are linked in a way that specifies how changes in one will lead to changes in another.

For example, consider the relationship between smoking and lung cancer. A statistical model can show that smoking is correlated with an increased risk of lung cancer, but it doesn't explain why this is the case. A causal model, on the other hand, can illustrate how smoking leads to biological changes in the body that increase the risk of developing lung cancer, thereby providing a more comprehensive understanding of the relationship.

In summary, the Reichenbach Principle underscores the necessity of seeking causal explanations behind observed correlations. While statistical models are powerful tools for identifying patterns and relationships in data, causal models offer a deeper level of understanding. They provide insights into the mechanisms and processes underlying these patterns, allowing for more accurate predictions, effective interventions, and a richer comprehension of the phenomena being studied.] [70]

Aspects of cognition	LLMs	PPL
Compositionality	×	✓
System 1 & 2		
Causality	×	✓
Analogy		
Abduction		
Symbolic Reasoning	×	✓

Table 1.1: title

11.1.6 Abduction

11.1.7 System 1 & 2

11.1.8 Concepts

from JB: LLMs learn how to complete sequences, they do statistics over patterns

Stephen shows that when LLMs create statistics over text, they first discover patterns, then syntax, then style, and only lastly semantics. Humans however start out by discovering semantics and then find patterns, syntax and lastly style.

- Can LLMs extract composable entities to recombine?

- could LLMs be regarded as symbolic like models? since each token is a vector and then the model builds aggregate representations given those vectors?

- with GFN we are trying to predict trajectories/ thoughts, rather than predictions.

Glossary

cognitive light cone . 58

Perception-Action Cycle . 61

Bibliography

[1] Stanislas Dehaene, Fosca Al Roumi, Yair Lakretz, Samuel Planton, and Mathias Sablé-Meyer. Symbols and mental programs: a hypothesis about human singularity. *Trends in Cognitive Sciences*, 26(9):751–766, September 2022.

[2] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pages 835–850, Virtual Canada, June 2021. ACM.

[3] Peter Blouw, Eugene Solodkin, Paul Thagard, and Chris Eliasmith. Concepts as semantic pointers: A framework and computational model. *Cognitive science*, 40(5):1128–1162, 2016.

[4] Tomer D. Ullman, Elizabeth Spelke, Peter Battaglia, and Joshua B. Tenenbaum. Mind games: Game engines as an architecture for intuitive physics. *Trends in Cognitive Sciences*, 21(9):649–665, September 2017.

[5] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017.

[6] Joshua S. Rule, Joshua B. Tenenbaum, and Steven T. Piantadosi. The Child as Hacker. *Trends in Cognitive Sciences*, 24(11):900–915, November 2020.

[7] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, February 2010.

[8] Karl Friston, Rosalyn J. Moran, Yukie Nagai, Tadahiro Taniguchi, Hiroaki Gomi, and Josh Tenenbaum. World model learning and inference. *Neural Networks*, 144:573–590, December 2021.

[9] Judea Pearl. Theoretical impediments to machine learning with seven sparks from the causal revolution. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, page 3–3, Marina Del Rey CA USA, February 2018. ACM.

[10] Arthur W Burks. Peirce's theory of abduction. *Philosophy of science*, 13(4):301–306, 1946.

[11] Johan Kwisthout. Most inforbable explanations: Finding explanations in bayesian networks that are both probable and informative. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 328–339. Springer, 2013.

[12] Maxwell JD Ramstead, Michael D Kirchhoff, and Karl J Friston. A tale of two densities: active inference is enactive inference. *Adaptive Behavior*, 28(4):225–239, August 2020.

[13] Daniel Kahneman. *Thinking, Fast and Slow.* Farrar, Straus and Giroux, New York, 2011.

[14] Philip Ball. Surfaces and essences: Analogy as the fuel and fire of thinking. *Nature*, 496(7446):424–425, 2013.

[15] S. Wolfram. *What Is ChatGPT Doing ... and Why Does It Work?* Wolfram Media, Incorporated, 2023.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[17] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building Machines That Learn and Think Like People, November 2016. arXiv:1604.00289 [cs, stat].

[18] Artur d'Avila Garcez and Luis C. Lamb. Neurosymbolic AI: The 3rd Wave. *arXiv:2012.05876 [cs]*, December 2020. arXiv: 2012.05876.

[19] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. *Program synthesis.* Number 4.2017, 1-2 in Foundations and trends in programming languages. Now Publishers, Hanover, MA Delft, 2017.

[20] Tomer D. Ullman, Noah D. Goodman, and Joshua B. Tenenbaum. Theory learning as stochastic search in the language of thought. *Cognitive Development*, 27(4):455–480, October 2012.

[21] Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. Robustfill: Neural program learning under noisy i/o. In *Proceedings of the 34th International Conference on Machine Learning*, page 990–998. PMLR, July 2017.

[22] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[23] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Building interpretable hierarchical knowledge representations with wake-sleep bayesian program learning. page 68.

[24] Nathanaël Fijalkow, Guillaume Lagarde, Théo Matricon, Kevin Ellis, Pierre Ohlmann, and Akarsh Potta. Scaling Neural Program Synthesis with Distribution-based Search, October 2021. arXiv:2110.12485 [cs].

[25] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27381–27394. Curran Associates, Inc., 2021.

[26] Yoon Kim, Chris Dyer, and Alexander Rush. Compound Probabilistic Context-Free Grammars for Grammar Induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy, July 2019. Association for Computational Linguistics.

[27] N. G de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae (Proceedings)*, 75(5):381–392, January 1972.

[28] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[29] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets, October 2022. arXiv:2201.13259 [cs, stat].

[30] Edward Hu, Nikolay Malkin, Moksh Jain, Katie Everett, Alexandros Graikos, and Yoshua Bengio. GFlowNet-EM for learning compositional latent variable models, February 2023. arXiv:2302.06576 [cs, stat].

[31] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to represent programs with graphs. *arXiv preprint arXiv:1711.00740*, 2017.

[32] Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The CLRS Algorithmic Reasoning Benchmark, June 2022. arXiv:2205.15659 [cs, stat].

[33] David Bieber, Charles Sutton, Hugo Larochelle, and Daniel Tarlow. Learning to execute programs with instruction pointer attention graph neural networks. (arXiv:2010.12621), October 2020. arXiv:2010.12621 [cs].

[34] Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Dudzik, Matko Bošnjak, Alex Vitvitskyi, Yulia Rubanova, et al. A generalist neural algorithmic learner. *arXiv preprint arXiv:2209.11142*, 2022.

[35] Han Peng, Ge Li, Yunfei Zhao, and Zhi Jin. Rethinking Positional Encoding in Tree Transformer for Code Representation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3204–3214, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[36] Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. Tree transformer: Integrating tree structures into self-attention. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, page 1061–1070, Hong Kong, China, November 2019. Association for Computational Linguistics.

[37] Qi He, João Sedoc, and Jordan Rodu. Trees in transformers: a theoretical analysis of the Transformer's ability to represent trees, December 2021. arXiv:2112.11913 [cs].

[38] Ke Wang, Rishabh Singh, and Zhendong Su. Dynamic neural program embedding for program repair. *arXiv preprint arXiv:1711.07163*, 2017.

[39] Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, Kaixuan Wang, and Xudong Liu. A Novel Neural Source Code Representation Based on Abstract Syntax Tree. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 783–794, Montreal, QC, Canada, May 2019. IEEE.

[40] Bruno C. D. S. Oliveira and Andres Löh. Abstract syntax graphs for domain specific languages. In *Proceedings of the ACM SIGPLAN 2013 workshop on Partial evaluation and program manipulation*, pages 87–96, Rome Italy, January 2013. ACM.

[41] Volker Peckhaus. Leibniz's Influence on 19th Century Logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2018 edition, 2018.

[42] Michael Rescorla. The Language of Thought Hypothesis. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2019 edition, 2019.

[43] Panu Raatikainen. Gödel's Incompleteness Theorems. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2022 edition, 2022.

[44] B. J. Copeland. *The Essential Turing*. Oxford University Press UK, 2004.

[45] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.

[46] Fosca Al Roumi, Sébastien Marti, Liping Wang, Marie Amalric, and Stanislas Dehaene. Mental compression of spatial sequences in human working memory using numerical and geometrical primitives. *Neuron*, 109(16):2627–2639.e4, August 2021.

[47] Artur d'Avila Garcez and Luis C Lamb. Neurosymbolic ai: the 3rd wave. *arXiv preprint arXiv:2012.05876*, 2020.

[48] Idan A. Blank. What are large language models supposed to model? *Trends in Cognitive Sciences*, 27(11):987–989, November 2023.

[49] Steven T Piantadosi. The computational origin of representation. *Minds and machines*, 31(1):1–58, 2021.

[50] Steven T Piantasodi and Felix Hill. Meaning without reference in large language models. *arXiv preprint arXiv:2208.02957*, 2022.

[51] Quan Do and Michael E Hasselmo. Neural circuits and symbolic processing. *Neurobiology of Learning and Memory*, 186:107552, 2021.

[52] Adam Santoro, Andrew Lampinen, Kory Mathewson, Timothy Lillicrap, and David Raposo. Symbolic behaviour in artificial intelligence. *arXiv preprint arXiv:2102.03406*, 2021.

[53] Douglas R. Hofstadter. *Gödel, Escher, Bach: an Eternal Golden Braid*. Basic Books Inc., 1979.

[54] Mark Blokpoel, HT Wareham, WFG Haselager, Ivan Toni, and IJEI van Rooij. Deep analogical inference as the origin of hypotheses. 2018.

[55] Jerry A Fodor. *The modularity of mind*. MIT press, 1983.

[56] Penelope A Lewis, Günther Knoblich, and Gina Poe. How memory replay in sleep boosts creative problem-solving. *Trends in cognitive sciences*, 22(6):491–503, 2018.

[57] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7(2):155–170, 1983.

[58] W. F. G. Haselager. Cognitive science and folk psychology: The right frame of mind. 1997.

[59] Stephen Palmer. Fundamental aspects of cognitive representation. 1978.

[60] Dagmar Zeithamova, Michael L. Mack, Kurt Braunlich, Tyler Davis, Carol A. Seger, Marlieke T. R. van Kesteren, and Andreas Wutz. Brain Mechanisms of Concept Learning. *Journal of Neuroscience*, 39(42):8259–8266, October 2019. Publisher: Society for Neuroscience Section: Symposium and Mini-Symposium.

[61] Chris Eliasmith. *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.

[62] Peter Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. The MIT Press, 03 2000.

[63] Anna Ciaunica, Michael Levin, Fernando E. Rosas, and Karl Friston. Nested Selves: Self-Organisation and Shared Markov Blankets in Prenatal Development in Humans. preprint, PsyArXiv, May 2023.

[64] David Vernon, Robert Lowe, Serge Thill, and Tom Ziemke. Embodied cognition and circular causality: on the role of constitutive autonomy in the reciprocal coupling of perception and action. *Frontiers in Psychology*, 6, 2015.

[65] Michael Levin. Bioelectric networks: the cognitive glue enabling evolutionary scaling from physiology to mind. *Animal Cognition*, May 2023.

[66] Michael Levin. Technological approach to mind everywhere: An experimentally-grounded framework for understanding diverse bodies and minds. *Frontiers in Systems Neuroscience*, 16, 2022.

[67] Michael Levin. The Computational Boundary of a "Self": Developmental Bioelectricity Drives Multicellularity and Scale-Free Cognition. *Frontiers in Psychology*, 10:2688, December 2019.

[68] Charlotte Caucheteux, Alexandre Gramfort, and Jean-Rémi King. Evidence of a predictive coding hierarchy in the human brain listening to speech. *Nature Human Behaviour*, 7(3):430–441, March 2023. Number: 3 Publisher: Nature Publishing Group.

[69] Hae-Jeong Park and Karl Friston. Structural and functional brain networks: from connections to cognition. *Science (New York, N.Y.)*, 342(6158):1238411, November 2013.

[70] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, May 2021.

1 Domain Specific Language (DSL)

1.1 Semantics

```
    semantics = {
    "empty": [],
    "cons": _cons,
    "car": _car,
    "cdr": _cdr,
    "empty?": _isEmpty,
    "gt?": _gt,
    "le?": lambda x: lambda y: x <= y,
    "not": lambda x: not x,
    "max": lambda x: lambda y: max(x, y),
    "min": lambda x: lambda y: min(x, y),
    "if": _if,
    "eq?": _eq,
    "*": _multiplication,
    "+": _addition,
    "-": _subtraction,
    "length": len,
    "0": 0,
    "1": 1,
    "2": 2,
    "3": 3,
    "4": 4,
    "5": 5,
    "range": _range,
    "map": _map,
    "iter": _miter,
    "append": lambda x: lambda l: l + [x],
    "unfold": _unfold,
    "index": _index,
    "fold": _fold,
    "is-mod": lambda x: lambda y: y % x == 0 if x != 0 else False,
    "mod": _mod,
    "is-prime": _isPrime,
    "is-square": _isSquare,
    "filter": lambda f: lambda l: [x for x in l if f(x)]
}
```

1.2 Primitive Types

```
    primitive_types = {
    "empty": List(t0),
    "cons": Arrow(t0, Arrow(List(t0), List(t0))),
    "car": Arrow(List(t0), t0),
    "cdr": Arrow(List(t0), List(t0)),
    "empty?": Arrow(List(t0), BOOL),
    "max": Arrow(INT, Arrow(INT, INT)),
    "min": Arrow(INT, Arrow(INT, INT)),
    "gt?": Arrow(INT, Arrow(INT, BOOL)),
    "le?": Arrow(INT, Arrow(INT, BOOL)),
    "not": Arrow(BOOL, BOOL),
    "if": Arrow(BOOL, Arrow(t0, Arrow(t0, t0))),
    "eq?": Arrow(INT, Arrow(INT, BOOL)),
    "*": Arrow(INT, Arrow(INT, INT)),
    "+": Arrow(INT, Arrow(INT, INT)),
    "-": Arrow(INT, Arrow(INT, INT)),
    "length": Arrow(List(t0), INT),
    "0": INT,
    "1": INT,
    "2": INT,
    "3": INT,
    "4": INT,
    "5": INT,
    "range": Arrow(INT, List(INT)),
    "map": Arrow(Arrow(t0, t1), Arrow(List(t0), List(t1))),
    "iter": Arrow(INT, Arrow(Arrow(t0, t0), Arrow(t0, t0))),
    "append": Arrow(t0, Arrow(List(t0), List(t0))),
    "unfold": Arrow(t0, Arrow(Arrow(t0, BOOL), Arrow(Arrow(t0, t1), Arrow(Arrow(t0, t0),
    ↪ List(t1))))),
    "index": Arrow(INT, Arrow(List(t0), t0)),
    "fold": Arrow(List(t0), Arrow(t1, Arrow(Arrow(t0, Arrow(t1, t1)), t1))),
    "is-mod": Arrow(INT, Arrow(INT, BOOL)),
    "mod": Arrow(INT, Arrow(INT, INT)),
    "is-prime": Arrow(INT, BOOL),
    "is-square": Arrow(INT, BOOL),
    "filter": Arrow(Arrow(t0, BOOL), Arrow(List(t0), List(t0))),
}
```

2 Experiment Hyperparameters

Hyperparameters	**Value**
α	0.3
β	0.7
γ	10
δ	150
ϵ	0.3
`replay_prob`	1 / 0.3
`fantasy_prob`	1
`learning rate: generative model`	0.0001
`learning rate: forward policy`	0.0001
`e-steps`	2.000
`m-steps`	2.000
`epochs`	5
`batch size`	4
`inference steps`	100

Table 2: Hyperparameters of both experiments.

3 Model Parameters

Class	Parameter	Value
IOEncoder		
	size_max	10
	d_model	512
RuleEncoder		
	d_model	512
Generative Model		
	d_model	512
	num_heads	8
	num_layers	2
	dropout	0.1
Forward Policy		
	d_model	512
	num_layers	2
	activation function	ReLU
Z		
	d_model	512
	num_layers	2
	activation function	ReLU

Table 3: Model Parameters

4 Formal Grammars

Context-Free Grammars (CFGs) are essential in defining the syntactical structures of many formal languages. We can formalize the notion of CFGs as follows:

Let $G = (N, \Sigma, P, S)$ be a Context-Free Grammar, where:

- N is a finite set of non-terminal symbols.
- Σ is a finite set of terminal symbols with $N \cap \Sigma = \emptyset$
- P is a finite set of production rules, where each rule is of the form $N \to (N \cup \Sigma)^*$
- S is the start symbol, with $S \in N$

Given such a CFG, the derived sentence space $\Pi(G)$ is the set of all possible strings (or sequences of symbols) derivable from S.

Given a Context-Free Grammar G and a defined objective function f that maps any program $p \in \Pi(G)$ to a real value representing its desirability or fitness:

Find p^* such that:

$$p^* = \arg \max_{p \in \Pi(G)} f(p)$$

In other words, the problem is to locate a program p^* within the vast program space $\Pi(G)$ defined by G that maximizes (or, alternatively, minimizes) the objective function f.

A **Probabilistic Context-Free Grammar (PCFG)** is an extension of a CFG G, denoted as $G' = (N, \Sigma, P', S)$, where:

- N and Σ are as defined in the CFG.
- P' is a set of production rules, where each rule $A \to \alpha$ is associated with a probability $P(A \to \alpha)$, representing the likelihood of selecting that particular rule. These probabilities are subject to the condition that, for each non-terminal A, the sum of probabilities for all rules $A \to \alpha$ is equal to 1.

5 Levenshtein Distance

Given two strings s and t of lengths m and n respectively, the Levenshtein distance $d(s, t)$ is defined as the cost of the cheapest sequence of edits needed to transform s into t. The Levenshtein distance can be efficiently computed using dynamic programming. The idea is to construct a matrix where each cell (i, j) represents the cost of transforming the first i characters of s into the first j characters of t.

The formula for filling in the matrix is:

1. If $i = 0$, then $d(i, j) = j$ (cost of adding j characters).
2. If $j = 0$, then $d(i, j) = i$ (cost of deleting i characters).
3. Otherwise:

$$d(i,j) = \min \begin{cases} d(i-1,j) + 1 \\ d(i,j-1) + 1 \\ d(i-1,j-1) + \text{cost}(s_i, t_j) \end{cases}$$

   where $\text{cost}(s_i, t_j)$ is 0 if $s_i = t_j$ and 1 otherwise.

The value of $d(m, n)$ will then be the Levenshtein distance between s and t.

6 Tasks