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

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.

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

2 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 [3, 4] [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 [5]. 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 [3, 4].

2.1 LLMs/ Self-attention / transformers

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

In his recent book "What is ChatGPT doing?", Stephen Wolfram analyses the internal mechanisms of ChatGPT [6].

It would be remiss to not describe and analyze large language models (LLMs) such as ChatGPT [source], Bard [source], and Llama [source], given their recent undeniable and phenomenal successes. These models are descendents of the Transformer architecture, originally introduced in 2017 by Vaswani et al. [7] 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

.

[in order to try to be self-contained,] I will briefly outline the inner workings of these neural network architectures.

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. Akin to Gördenford's conceptual spaces **??**, tokens are essentially laid out in a high-dimensional space with the intention of tokens being organized according to their semantic meaning. E.g. "dog" should be closer to "canine" than to "panda". Various levels of embeddings are encoded, which gives aggregate words, sentences, paragraphs, etc. positions in this latent space.

At the heart of the Transformer lies the self-attention mechanism, which enables the model to weigh the significance of different tokens in a sentence relative to a given token.

Formally,

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

where:

Q = A matrix of query vectors, where each row represents a query for a single word in the input sequence.

K = A matrix of key vectors, where each row represents a key for a single word in the input sequence.

V = A matrix of value vectors, where each row represents a value for a single word in the input sequence.

d_k = The dimension of the key and query vectors.

softmax = A function that normalizes the attention scores such that they sum to 1.

The significance of each part of the formula is as follows:

Q, K, and V: These matrices are obtained by projecting the input sequence into three different spaces using three separate weight matrices. This projection allows the model to learn different representations of the input sequence, which can be useful for different tasks. $\frac{QK^T}{\sqrt{dk}}$: This term calculates the similarity between each query vector and each key vector. The dot product of two vectors is a measure of their similarity, but it is also sensitive to the scale of the vectors. Dividing by the square root of the dimension of the key vectors helps to normalize the attention scores such that they are comparable across different words in the input sequence. softmax: The softmax function normalizes the attention scores such that they sum to 1. This ensures that the attention weights represent a probability distribution over the words in the input sequence. V: The value matrix contains the representations of the words in the input sequence that will be attended to. The attention weights are multiplied by the value matrix to produce a weighted sum of the value vectors. This weighted sum represents the context-aware representation of the current word in the input sequence. Self-attention is a powerful mechanism that allows models to learn long-range dependencies in input sequences. This is because the attention weights can be used to focus on different parts of the input sequence, depending on the context of the current word.

Multi-Head Attention: Instead of a single set of attention weights, the model uses multiple sets, enabling it to focus on different parts of the input for different tasks simultaneously. Feed-forward Networks: Each layer of the transformer contains a feed-forward neural network, which is applied independently to each position. [This allows the transformer to make statistics over compounded sentences (does wolfram explain it?)]

The *encoder* is a block that takes an input sequence of tokens and produces a sequence of hidden states. The hidden states represent the information that the encoder has extracted from the input sequence.

The *decoder* is a neural network that takes the hidden states from the encoder and produces an output sequence of tokens. The decoder is trained to predict the next word in the sequence, given the previous words and the hidden states.

Next token prediction.

In his book, Wolfram shows 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", or jabberwocky sentences, show that there is more to language than syntax. There seems to be a semantic grammar, which LLM models learn implicitly.

Because LLMs have seen vast amount of sentences of the structure if X then Y Logic is a way of saying that sentences that follow certain patterns are reasonable, while others are not. For example, it's reasonable to say "All X are Y. This is not Y, so it's not an 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 by correlation and they don't "know" the actual rules of logic.

- we don't know yet whether implicit logic has actually been discovered or whether the model is just that good by correlation, since it has seen so much data.

This is the learnt semantic space. We can look e.g. at analogies such as man is to king as woman to queen.

When LLMs create a sentence, they are essentially following a trajectory in semantic space. This can be seen as a POMDP?

2.1.1 Limitations

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.

2.2 Compositionality

[8].

Humans possess a native capacity for meta-cognitive evaluation, instinctively gauging the reliability of their own knowledge. 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.

Humans are endowed with an intuitive grasp of causality, which functions as the underpinning for both predictive and counterfactual reasoning. This enables humans to extrapolate beyond the confines of empirical data, to conjure hypothetical worlds e.g. in imagination, play, and dreaming.

- Can LLMs extract composable entities to recombine?

2.3 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]

2.4 Causal Models

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) [ref from GFN]. The vertices in this graph denote variables, which, in this context, could be seen as cognitive states. The directed edges, meanwhile, signify causal relationships, pointing from cause to effect.

these directed edges often correspond to conditional probability distributions. For instance, if we have a directed edge from $X \to Y$, it implies that the distribution of Y is conditional on X, i.e. $P(Y|X)$. The model encompasses these conditional distributions for all variables given their parents in the DAG, encapsulating the joint distribution over all variables.

Causal models facilitate interventions, counterfactuals, and causal inference. Intervention is the ability to model the outcome of purposeful changes, represented mathematically as "do" operations [pearl]. 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? This provides a structured framework for hindsight, and one could argue, even a modicum of wisdom.

[refer to pearls paper of three levels of causality [9], go more in detail?]

Using GFlowNet we can make inferences about intermediate variables. This is because we approximate probability distributions and not point estimates.

2.5 System 1 & System 2

Kahneman describes two distinct systems that characterize the dual aspects of human cognition [source]: *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. This system 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. It can be equated to the recognition density in Bayesian inference. 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.

- example: when learning to play piano, we need conscious effort on certain phrases. Once chunked, we can use them to learn more complex phrases.

[8]

[how do we get those primitives? I think we can get them innately, through extraction, or perhaps else, so there may be multiple sources. this means that we dont only improve the dsl internally, but perhaps also through communication (memes) or further extraction; however, don't mistake the explanation of something as the actual thought. you can never see how to actually construct a thought. but concepts are symbols and people can show you the symbolic trajectory you need to take to get to the result.]

2.6 Language of Thought

Jerry Fodor suggests that thought takes place within a mental language - sometimes referred to as "Mentalese." Under this hypothesis, our cognitive processes can be viewed as 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. Fodor's theory implies an innate structure underlying human cognition, with systematic, rule-governed operations that manipulate symbols.

Keep this here or in discussion?

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

Under the Free-Energy Principle [necessary?, maybe just Bayesian principles in general], organisms are seen to construct internal models of the internal and external world to predict and hence reduce the surprise of sensory inputs, which requires a similarly nested, hierarchical organization of cognitive processes [10, 11].

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 generating predictions. What ascends is the prediction error, the mismatch between predictions and actual [sensorimotor, proprioceptive, interoceptive, etc.] input. In this framework, perception becomes an query-response mechanism. The brain tries to infer probable causal factors from its received data.

3 Bayesian Inference

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:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Where:

- $P(H|E)$ is the posterior probability of hypothesis H given evidence E.
- $P(E|H)$ is the likelihood of evidence E given that hypothesis H is true.
- $P(H)$ is the prior probability of hypothesis H.
- $P(E)$ is the marginal likelihood, probability of evidence E.

In this Bayesian framework, the joint probability $P(E, H) = P(E|H) \cdot P(H)$ is referred to as the generative model that hypothesizes how sensory data are generated by the hidden states of the world. 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 $Q(H|E) \approx P(H|E)$ the brain's inference about the state of the world given the data. [explain marginalisation problem?]

3.1 Game Engine in the Head

- game engine in the head
- Reverse engineering the world
- Intuitive physics etc.
- Extension of the LOT, circumventing the downfall of GOFAI, frame-problem, etc.

4 Probabilistic Programming

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 [12]. 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. 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.

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] [13]. 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].

[the assumption we make here is that features are somehow aggregated or extracted into symbols [see [14]], primitives, along with possibly innate symbols [8]]

I am following the assumption that we don't start from a blank slate but from some initial concepts, here operationalised as primitives, fundamental program components. These initial primitives comprise a minimal domain-specific lanugage (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. In the following, I will first introduce the state-of-the-art model in this regard in detail, to present the possibilities of this framework as well as the methods used, many of which inspired my own model.

5 Methods

5.1 DreamCoder

[which supercede previous models, e.g. RobustFill, DeepCoder, etc.] 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 [1]. 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 [15] [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 [16] 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 1(A) shows examples of tasks across different domains. Figure 1(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.

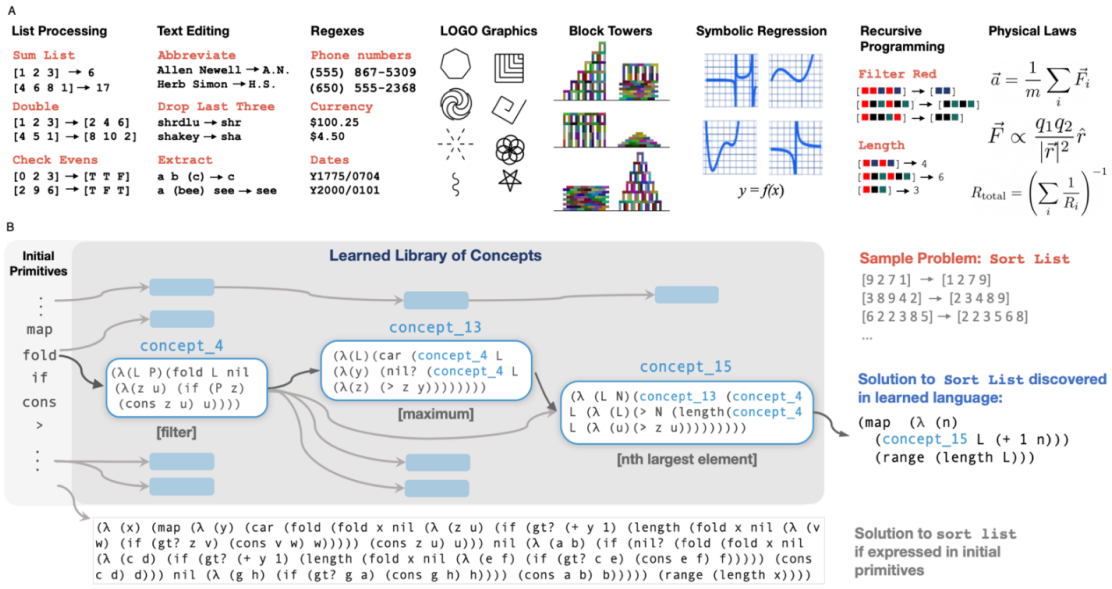

Figure 1: (A) 8 different domains of tasks. (B) Representation of the learned library of concepts. On the left we see the initial primitives from which the concepts in the middle region are composed. On the right we see a task as input-output relations and the found solution. On the bottom is the same solution expressed only with initial primitives. Image taken with permission from the original paper [1].

5.2 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 [17].

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 (such 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 the same programs can occur. An additional benefit of sampling over search is space efficiency - already created programs don't need to be memorised (with the possible downside of resampling).

5.3 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 [18]. 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.

6 FlowCoder

In the following I will describe the DeepSynth framework, the DreamCoder architecture, and GFlowNets in detail and point out how I utilised each component to build my model, FlowCoder.

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. Named variables are replaced with numeric indices representing the number of enclosing lambda abstractions that bind the variable. See figure 2 for a visualisation of an AST using deBruijn indexing.

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

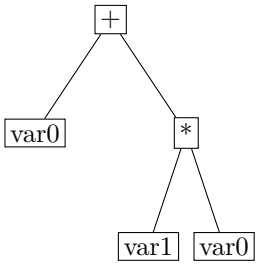

Figure 2: An example of an abstract syntax tree (AST) using deBruijn indexing. This translates to `var0 + var1 * var0`.

Formally, the overall objective of DreamCoder is to find the optimal DSL $\mathcal{D}$, which is a set of typed λ-calculus expressions, along with optimal weights θ to navigate it, such as to solve a given a set of tasks $x \in X$.

I.e. the objective is to find the joint distribution:

$$J(\mathcal{D}, \theta) = P(\mathcal{D}, \theta) \prod_{x \in X} \sum_{\rho} P(x|\rho) P(\rho|\mathcal{D}, \theta)$$

where:

$P(\mathcal{D}, \theta)$ = Prior distribution over languages and parameters
$P(x|\rho)$ = Likelihood of task $x \in X$ given program ρ

Since computing $J(\mathcal{D}, \theta)$ entails marginalising over all programs, which is intractable, they instead marginalise over a finite beam [definition] per task $\mathcal{B}_x$ which is operationalized as a lower bound on the joint density. This bound is even tighter than the ELBO [reference].

$$\mathcal{L}(\mathcal{D}, \theta, \{\mathcal{B}_x\}) = P(\mathcal{D}, \theta) \prod_{x \in X} \sum_{\rho \in \{\mathcal{B}_x\}} P(x|\rho) P(\rho|\mathcal{D}, \theta)$$

The model operates in three distinct phases:

6.0.0.1 Wake In the wake phase, the objective is to find the best program ρ for each task $x \in X$ DreamCoder is presented with. Each task consist of one or a few (~ 10) examples. The objective is to maximize $\mathcal{L}$ w.r.t. to the beam $\{\mathcal{B}_x\}$

6.0.0.2 Sleep: Abstraction A key component of the success of DreamCoder is the refactoring of programs. In this phase, DreamCoder consolidates new abstractions from the programs it has found during the wake phase. Experiences are replayed, and commonalities between programs are found so that they can be abstracted into new concepts to use in the future. In the abstraction phase $\mathcal{L}$ w.r.t. $\mathcal{D}$ is maximized.

6.0.0.3 Sleep: Dreaming The objective during the dreaming phase is to train the recognition model Q to perform MAP inference by both fantasies, and replays. When fantasising, programs are drawn from the learned library as a generative model. The model is then trained on the produced tasks. This ensures a large and varied data-set to learn from and encourages out-of-distribution generalisation. Replays, simply training on previously seen task-program pairs, ensures that the model improves its predictions on the actual tasks it needs to learn and doesn't forget how to solve tasks it has already solved correctly.

In dreaming they maximize $\mathcal{L}$ w.r.t. θ.

$$\mathcal{L}_{\text{MAP}}^{\text{Replay}} = \mathbb{E}_{x\sim X}\left[\log Q\left(\arg\max_{p\in\mathcal{B}_x} P(p|x,D,\theta)\Big|x\right)\right]$$

$$\mathcal{L}_{\text{MAP}}^{\text{Fantasy}} = \mathbb{E}_{x\sim(D,\theta)}\left[\log Q\left(\arg\max_{p} P(p|x,D,\theta)\Big|x\right)\right]$$

The authors chose to optimize MAP rather than the posterior because together with their parameterization of the recognition model, this encourages to break syntactic symmetries.

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.

(see appendix 1)

It is apparent that the abstraction phase is crucial for learning the DSL. In this project however, I want to focus on learning a good sampler. The model can subsequently always be extended to include the abstraction phase and thus expand the DSL. Therefore, I am using DeepSynth's framework. 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. A prediction model is then 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. 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.

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

6.0.0.4 Syntactic constraints

Same constraints as DeepSynth: type: `int list -> int list`

6.0.0.5 Tasks The tasks I am using are input-output relations in the list editing domain, originally from DreamCoder. 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 to be solved given the DSL, so I additionally filtered those out. [See appendix for examples].

We say that a task is solved if a program is found which satisfies all examples. The timeout of 100s only takes into account the search time and the evaluation times and not the time to query the neural network for predictions.

- n dreamcoder Tasks, filtered like in DeepSynth to only have certain types.
- We work with list editing tasks (give a couple of examples)

6.0.0.6 Rules I am predicting rules in the CFG, rather than primitives. I.e. edges between two symbols.

So how can we build a good sampler?

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. 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. 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, such that each rule is unique. Thus, every predicted node (rule) has exactly one parent, in other words, I am essentially linearizing 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:

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

6.1 Expectation-Maximization

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 on the E step.

In the E-step I optimize the recognition model, i.e. the forward policy of the GFlowNet that is proportional to the posterior and to the reward function.

Here, I'm sampling real data as well as a trajectory and use the trajectory balance loss to update the parameters of the forward policy.

In the M-step I sample a trajectory and only use the reward to update the parameters of the generative model.

INCLUDE PSEUDOCODE

Solving the joint optimization problem is difficult. I'm taking inspiration from DC and GFN-EM to use wake-sleep to train the models.

6.1.0.1 Replay

In replay, I train the recognition model to not forget previously solved tasks.

6.1.0.2 Fantasy

[also generative model?] In fantasy I train the recognition model to solve out of distribution tasks, so it learns to extrapolate to unseen data.

- In sleep we are guiding the model with correct trajectoriees. - include consideration of how often and when to sleep.

[discuss here that we are using a transformer to represent a neural PCFG?]

7 Model

7.0.0.1 generative model

As we have seen in the Transformer part, self-attention is an algorithm which lets the transformer learn syntactic and semantic structure from its data. This inductive bias is stronger than the purely syntactical nature of the PCFG. [neural PCFG, etc.]

The GEN model is. a transformer

7.0.0.2 Recognition model

while the recognition model si a MLP on top of the transformer. The partition function Z is also parameterized sa a MLP ib top of the transformer.

7.0.0.3 IOEncoder

Task embedding The IOEncoder is tasked with encoding input-output (IO) pairs into continuous vector representations. It encodes an IO pair into a flat sequence with special tokens ('IN' and 'OUT') indicating the boundary between input and output. - Show the exact model (transformer, how many layers, etc. and make a table of that? or is it already in the table?)

- explain how tasks are created (and that it also takes time if we create new tasks from scratch for replay and fantasy.)

from Deepsynth[Input encoding Each list in the examples is encoded in a naive way by mapping each element of Lin to [0, 60]. We additionally use two special symbols PAD and NOPAD, leading to a fixed size encoding of a list into a vector of size 2Lmax. Hence an example is encoded as a tensor of shape (ninputs,2Lmax) where ninputs is the number of inputs in the example.]

- I deviate from the DeepSynth and DreamCoder implementation to make it more suitable for a Transformer. Therefore, I ...

See 1 for parameter details (i think there are more, check it.).

7.1 RuleEncoder

State embedding - DSL and syntactic constraints are compiled into a CFG The RuleEncoder is designed to transform programming rules derived from a Context-Free Grammar (CFG) into continuous vector representations, or embeddings. in order to have semantics and not just syntax, i.e. use additional information of the CFG, I create a neural PCFG. (see kim) In each step, I predict the next rule from the cfg, until a trajectory is done. This means that a state is a trajectory of rules and a program is represented as such a list. 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].

since im using the cfg to extract only the allowed rules, it was easier to embed rules rather than primitives,

Rules could be encoded including additional information like tree depth, arg idx, parents, siblings, etc. since an AST is two dimensional, there are many expansion orders. my model is a transformer with a MLP on top to predict logits for the rule. I could have another, to predict the position to attach the next node. Other than that if i only embed primitives instead of rules, We could use a GNN and do both node and edge prediction. Otherwise, we also need to decide expansion order since rules are context free, i.e. if we only predict the rule there may be multiple parts of the tree where this could be applied to. However, there may be benefits in expanding in different orders. We could apply a second GFN to predict the next best node to expand. In the cfg we have terminals and nonterminals

When constructing an AST, it can be expanded in various orders. E.g. we could predict the tree bottom up, i.e. predict terminal nodes and then connect them progressively. The benefit of this is that we can evaluate partial expression. However, once a partial tree is predicted, the other arguments of the function are dependent on it so we have to keep expanding nodes upwards until ...

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.

Moreover, in the original GFN, as explained above [ref] we need a prediction for the backward logits. however, since we are creating trees top-down, every state has exactly one parent. therefore, we can make the calculations simpler and omit the backward logits. In general, it might not be a bad idea to include it since we can use it to do other kinds of inferences [e.g. in using subtrajectory balance].

1. We get a task (how do we sample?)
2. the task gets encoded
3. goes through transformer,
4. GFN predicts a rule, which is now the next state
5. we repeat this until we have a program
6. construct the program from the array of rules
7. evaluate program against ideal output

8. calculate TB loss
9. Update model

1. same as above, but different loss. update generative model rather than

- Data augmentation
- Self-supervised learning
- Sleep phase
- Replay
- Fantasy
- Overfitting
- Loss optimality. Quality vs Quantity
- Explain CFG, parameters like program depth, sizemax etc.
- How many programs can actually be created?
- Exploration
- What is the generative model? The CFG or the Transformer?
- Minimum description length (how is that achieved in DC (MAP?) and DS(Expectation of number of programs created before a solution is found)?)
- syntactic symmetries, symmetry breaking, show DC MAP vs POST, avoid adding 0, mult 1, etc.
- In the paper they say they produce programs up to depth 6
- We want to find a probability distribution that is proportional to a reward.
- Explain the parameterization. I.e. we could parameterize the rules vs the primitives.
- Write about the marginalization. Can we marginalize the CFG? How does DC do it? (they only marginalize over a beam)
- Do we want multiple solutions? MAP vs MLE + how could we convert one to another (Talk also about DC, and why they chose MAP)
- Surfaces and Essences analogy example abc:xyz :: abd: (wyz/ xya)
- Neural PCFG (Compare to Neural PCFG from Rush and GFN-EM)
- what is the difference between amortized EM and GFlowNet EM?
- I think GFlowNet-EM is using a neural PCFG as a generative model.
- write about tractability
- should there be one or multiple world models? what is the world model here? the PCFG? i.e. the generative model?

- Write about context free grammars. number of possible derivations.
- learning your own dsl
- should there be one dsl or multiple?
- it could be e.g. having multiple representations/ models of the world and then a model on top of that which tells you which model is useful. This reminds me of society of mind and also of neural darwinism. World models may be competing with one another.
- Sticking with a task for how long, before switching
- Evaluation of variables
- Now we are predicting rules, and encoding them individually. We could also use their features like depth, arg idx etc. to encode them
- Why we dont need backward logits
- when we only predict rules, it may be ambiguous. The same rule could expand different parts of the tree. That's why DFS
- we want to show that using the transformer for semantics makes sense, therefore we need to show that it learnt program embedding. relating tasks to programs and also relationships between tasks and relationships between programs. Also relate that to the embeddings of tasks in DC, which shows that similar tasks are approximately in the same space.
- Explain what exactly this task shows or is supposed to represent. Algorithmic thinking, or any type? Dehaene, (also sensory data pattern prediction machine paper)
- explain what the batches do
- I embed all the rules (check this with neuralPCFG and maybe embed the cfg so that you have a better generative model) (not the depth, arg number etc.)
- I am searching for the posterior (aligning with active inference), because i want diverse solutions, i.e. i want to learn the solution space for each task, multiple modes, not just the max mode. (how does this compare to MDL)
- does the model align with FEP
- If the CFG is generative, is that not a causal model? can there be a generative causal model? be clear about those terms.
- What do we need? Differently from the other models which do not embed programs (they do have a programencoder, so what is that then? the output). The output tensor is encoded as a program. The programs themselves are never embedded. What is the difference?
- Reasons for using a GFlowNet
- Transformer allows for semantic relationships (although we still don't use control or data flow, but we could in principle)
- We want to approximate a multimodal distribution, unlike DreamCoder in which the MAP estimate refrains from finding semantically equal but syntactically different solutions.

- Implicit vs explicit generative model, neuralPCFG, embedding rules.
- MAP vs (convergent vs divergent thinking.)
- One shot (predicting bigram transition matrix) vs trajectory (predicting logits in each step)
- argument for attention and why it might be necessary to capture semantics.
- transformers learn nested relationships (see chapter wolfram) (relationships of increasing complexity)
- Nathaniel compare enumeration based methods with sampling methods. quality vs quantity.
- If the sampler is good enough, there is no need for search
- does anything prevent DC from training further, improving the recognition model, making it a sampler? (it doesn't take partial programs, it only takes tasks.)
- We have seen that if a GFN is trained well enough, it should act as a really good sampler. So we use that

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

Algorithm 1 Function get_next_rules

```
function GET_NEXT_RULES(self, S, depth)
    return {(S, p) | p ∈ self.cfgs[depth].rules[S].keys()}
end function
```

Algorithm 2 Function get_mask

```
function GET_MASK(self, rules)
    mask ← [1 if rule in rules else 0 for rule in self.model.state_encoder.rules]
    mask ← torch.tensor(mask, device=device)
    return mask
end function
```

Algorithm 3 Function sample_program_dfs

```
function SAMPLE_PROGRAM_DFS(self, batch_IOs, depth, ϵ = 0., β = 1.)
    Initialize states, total_forwards, final_logZs, frontiers, programs
    while any(frontiers) do
        Calculate forward_logits, logZs
        Handle exploration and tempering
        for i ← 0 to len(batch_IOs) − 1 do
            if frontiers[i] then
                Handle rule sampling and state update
            end if
        end for
    end while
    if not any(frontiers) then
        Reconstruct final programs
        return final_logZs, total_forwards, reconstructed, states
    end if
end function
```

7.1.1 Reward

In order to train the model, we need an informative reward, i.e. it should provide a gradient. Of course we cannot just use the actual program to compare it with. In deepsynth they train by using the cross entropy loss between the nueral network aoutput and an encoding of the solution program. In reality we do not have the correct solution [think about this. we do also learn from others. so its not entirely out of question.]. Instead, we can only compare the real and predicted output. However, there are a few methods we can try to turn the comparison of sequences into a reward. Since we are working with numeric list editing, ... After some initial experiments of using a Hamming distance, comparing embeddings using cosine similarity, mutual gain, I decided to use an edit distance, more specifically, Levenstein edit distance (which is continuous rather than discrete.)

Another idea is to use an energy based model, which essentially is a model that learns to predict the reward. however, this increases the computational overload and was not necessary for this toy problem.

The edit distance, often referred to as the Levenshtein distance, 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. 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.

Since the Levenshtein distance returns a discrete value, I normalize it over the max length of the sequences, and since there may be more than one example per task, I average it over all examples. Quantifying the result in this way gives us quite a good estimate of the model's capabilities. E.g. when we have two lists [1,2,3] [1,2,3] ...

- The model was trained on the filtered DC tasks
- with a random 50 50 train test split
- batch size 4
- syntactical constraints

- GPU NVIDIA Tesla A100
- 2000 e-m steps
- with thresholding
-
- In inference, the model still has batchsize 4
- syntactic constraints
- we give it a 100 inference steps (400 programs are created)
- run on all 145 tasks
-

7.2 Results

7.2.1 Training

- How many programs are solved?
- Diverse solutions?
- did the model converge?
- how long did it take to find a program?
- what are the other programs created?

inference: - (especially when the model solved extrapolated tasks, were these programs found before in training?) - how long to solve tasks? - if solved, how often? - which other programs were created when trying to solve a task?

- Speed/ how long did it take to solve the tasks
- How long does it take to solve a program?
- Which tasks are solved
- Which tasks are not solved, why?
- Comparison with deepsynth
- How many programs are uniquely created? vs solved.
- Make a histogram of all the programs. Can we make it over time? I.e. for each program we should see how often it is created over time.
- The program is not loss optimal, i.e. we may get stuck in a local minimum
- Compare to baseline (Uniform PCFG?)
- my hopes would be that similar programs are embedded close in space. - how can we check that?

- visualize task embeddings (from DC) and the according programs. Maybe take the programs with the highest reward even if it didn't solve it
- Visualize rule embeddings. what would i expect?
- Training vs. inference (check the speed of finding solution before and after training and compare. Is the inference really fast? then it means that in principle we can get a good approximation to the reward)
- In Deepsynth they train on 10.000 examples from a random PCFG. then the model is shown the examples once, and they search through the PCFG for n seconds until they hopefully find the correct program. the problem is that the randomly created programs suck. they don't resemble real world programs in the slightest. if i train the network directly on the tasks, then yeah, were training it first which should be amortized and then inference should be fast.
- Check the DC paper. What can we actually compare?

We can see that even after several training rounds, the model still explores alternative solutions. This indicates. The human analogue is divergent thinking.

7.2.2 Inference

7.2.3 Embeddings

We can visualize embeddings using t-SNE (t-distributed Stochastic Neighbor Embedding). The goal of t-SNE is to group similar points together and separate dissimilar ones. If certain tasks are clustered together on the plot, this suggests that the model finds them to be similar in some way.

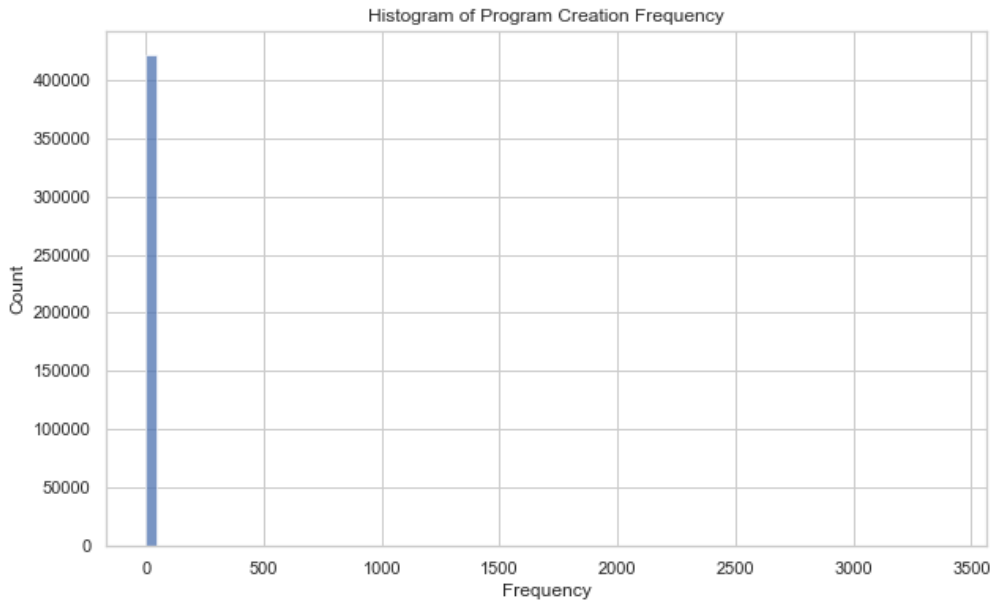

Figure 3:

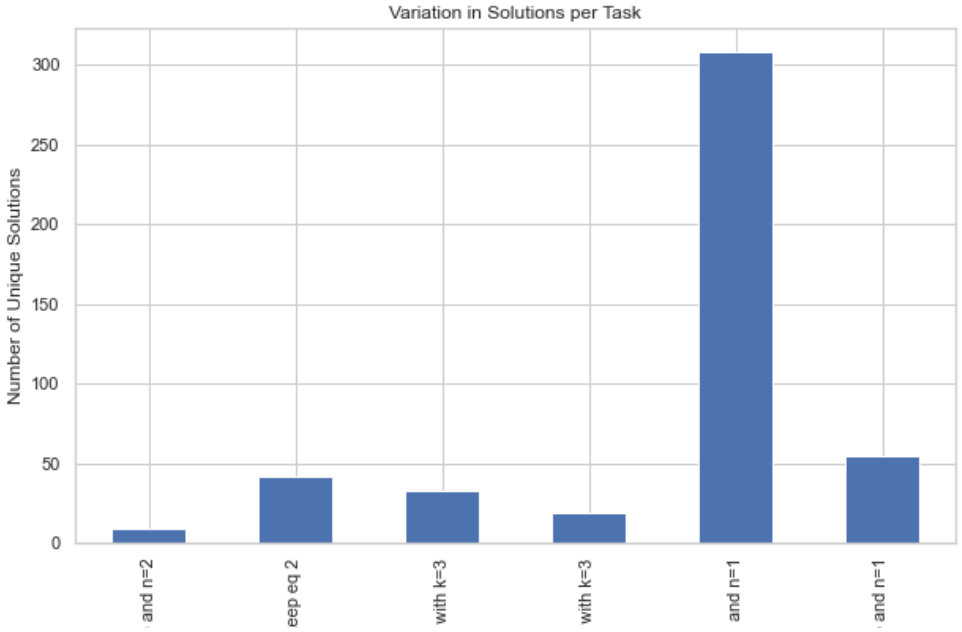

Figure 4:

Figure 3 presents a histogram illustrating the frequency distribution of program creations across various tasks. This visualization aids in understanding the commonality of specific programs in solving different tasks, highlighting the most frequently utilized programs. The x-axis represents the frequency of program creation, while the y-axis indicates the count of occurrences within the dataset.

Figure 4 showcases a bar chart detailing the diversity of solutions per task. Each bar corresponds to a specific task, with the height representing the number of unique solutions that successfully solved the task. This chart effectively conveys the variability and complexity of solutions across different tasks, illustrating tasks that exhibit a higher degree of solution diversity.

In Figure 5, a bar chart is used to depict the solution rate for each task. The tasks are displayed along the x-axis, and the corresponding solution rates are plotted on the y-axis. This chart provides a clear comparison of the relative difficulty or tractability of each task within the dataset, as indicated by the proportion of successful solutions.

Figure 6 displays a histogram of the reward distribution across all attempts. This graph offers insights into the typical reward outcomes, including the most common reward values and the spread of rewards. The x-axis indicates the reward magnitude, while the y-axis shows the frequency of each reward range, highlighting the central tendency and variability of the reward structure.

Figure 7 is a line chart that explores the relationship between the depth of the programs and their success rates. The depth of a program is plotted on the x-axis, and the corresponding success rate is indicated on the y-axis. This analysis is crucial for understanding how the complexity of a program, as measured by its depth, affects its likelihood of successfully solving a task.

In Figure 8, two line charts are presented, one for the success rate over different epochs and the other for the success rate over a number of steps. These charts elucidate the progression of success rates over time (epochs) and the iterative solving process (steps), providing a dynamic view of the

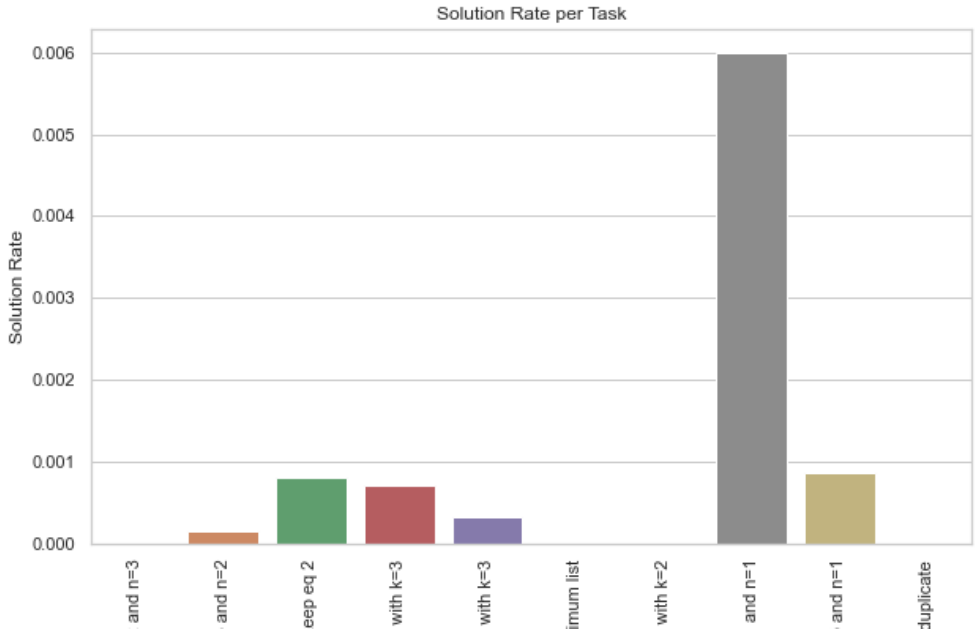

Figure 5:

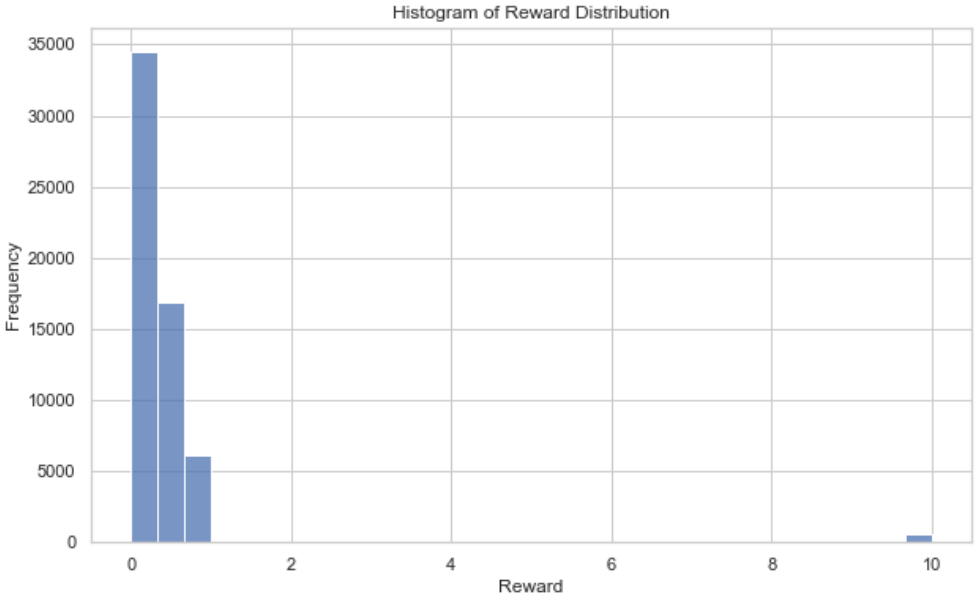

Figure 6:

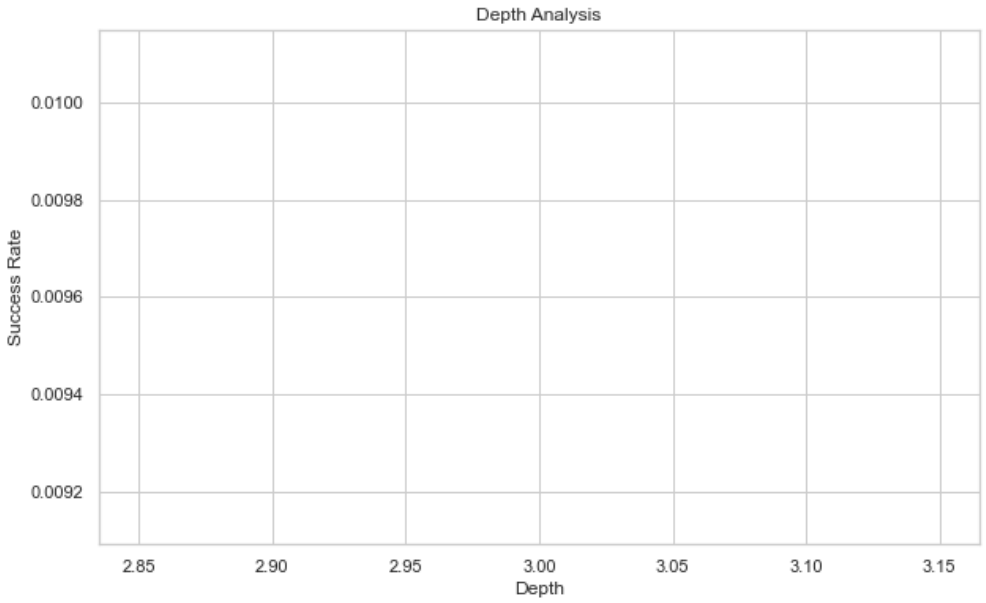

Figure 7:

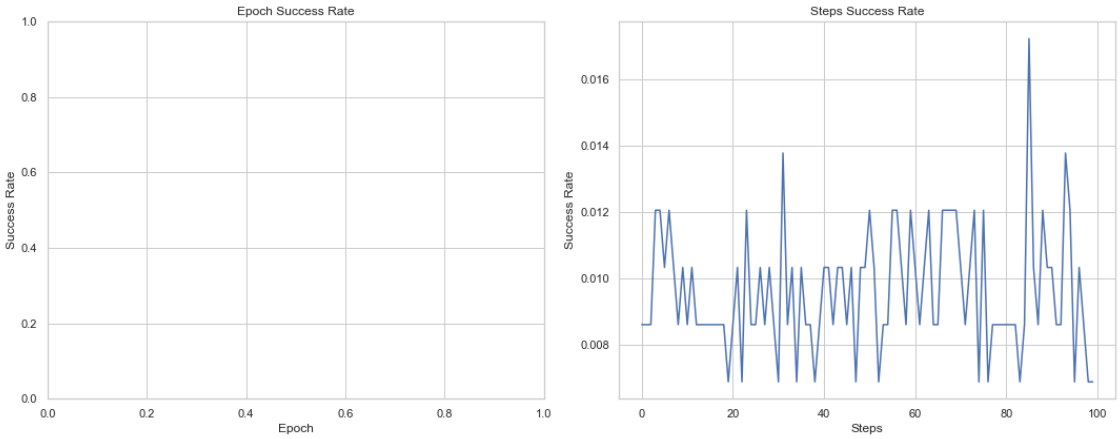

Figure 8:

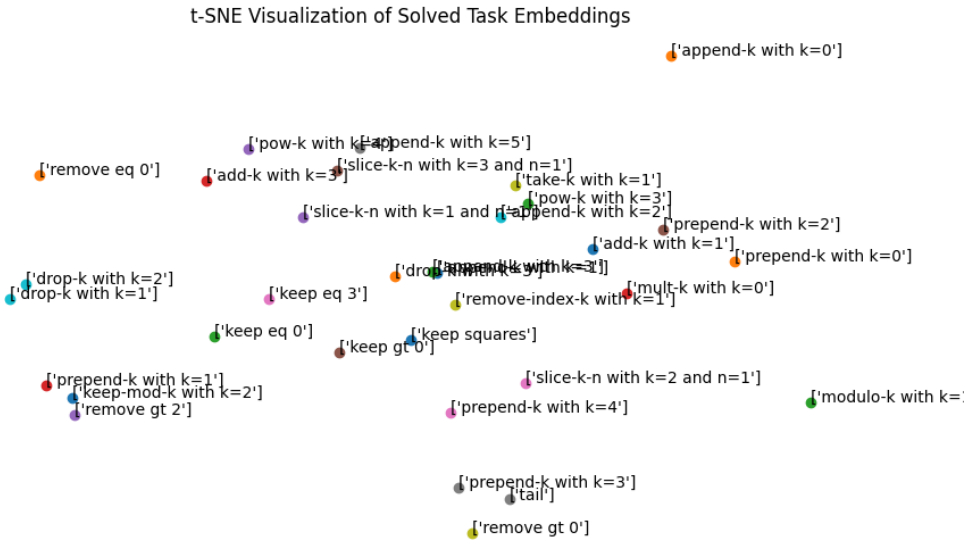

Figure 9:

learning or problem-solving efficiency.

7.3 Discussion

Here a CFG is used rather than the more dynamic but computationally heavy inference of variable scoping rules [explain] to determine permitted rules.

discuss that we actually need to infer the CFG but for computational reasons we just use the CFG. Otherwise, we have to find out which rules are allowed since we are dealing with types. Now is there a biologically plausible equivalent to types or is it completely a constraint of the method we are using? I suspect that there are no types. It's all dynamic. But that's okay for now.

Time comparison It is somewhat difficult to compare my method with other approaches. other approaches parallelize processes on multiple CPUs (20-100 CPUs in DC). It takes DC 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) - they show that the refactoring algorithm (which i havent used) helps a lot in list processing tasks. - In DeepSynth, they only count the time of search, not of querying the model. - In Gflownet-EM (and other GFN papers) they train on over 50.000 epochs for simple problems. So this approach takes long to converge, but we reap the benefits at inference time.

- discuss the variable/ non variable batching. (credit assignment.)
- mode seeking. difficulty of broad and focussed modes.
- MAP want to find the best (and shortest, given the right inductive bias) solution to the program. MLE tries to find a posterior distribution, i.e. all solutions of the task. (syntactically

different but semantically equivalent solutions. Why do you think this would be necessary in real life. Argue.)

- Perhaps in this toy scenario we don't necessarily need many solutions, but in general we do.
- 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
- Temporal dimension? (GLOM includes time)
- in DeepSynth and DreamCoder, because of outsourcing the weights to the PCFG, they can parallelise it to CPUs rather than using more expensive GPU time.
- Talk about batches and how it might speed up the process but perhaps make it more difficult to learn?
- 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?
- infer the cfg rather than have it as given. in DC they use variable scoping rules to determine which primitives are allowed, in deepsynth they just create a cfg because its faster.
- Reward should be learnt (Energy based model). Where does the brain get it from?
- Is Levenshtein a reasonable reward? think about computational complexity (Kologorov complexity)

7.3.1 Complexity Analysis

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 the 'GFlowNet' model 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. However, this worst-case scenario is likely to be rare in practical applications, where both d and k are bounded and relatively small. In these experiments only one argument per function is allowed.

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.

7.3.2 Limitations

7.3.3 Future Work

- 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.

7.4 Conclusion

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

7.5 Discussion

- 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?
- What have we learnt from autopoeisis, FEP, self-organisation, etc. Is my general approach a viable explanans to the eplanadum in question: is the conceptual world constructed by the same process we see in nature etc.
- If so, are programs a good representation?
- What are the general pitfalls of symbols? (maybe this should go in general discussion)
- other models like GNN, other AST representations, other spaces, hyperbolic, etc.

7.5.1 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 [definition?] 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:

$$\text{ELBO} = E_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)||p(z|x)) \tag{10}$$

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

- **Hierarchical Models**: Both probabilistic programming and human cognition exploit hierarchical structures, where higher levels capture more abstract representations and lower levels deal with details. This facilitates learning and transfer of knowledge across different tasks and contexts. [introduce Compositional LVMs]

- **Active Inference**: Probabilistic programming in cognition implies that the brain is engaged in active inference, selecting actions based on their expected outcomes to minimize the divergence between predicted and actual states (akin to minimizing a cost function in a program). [we are learning an action policy]

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] [19]. [include pros and cons]

[20]

ASTs [21] [22]

[related work section]

7.5.2 Biological Plausibility

- Why are/ aren't types plausible? Related to categories?
- CFG? where does it come from?
- primitives?
- How are programs represented (ASTs, GNNs, other ideas from different papers)
- Tree-Transformers
- Loss optimality
- There is a trade-off between finding many programs rapidly, or fewer programs that are more likely to solve the problem.
- enumeration vs sampling
- They show that all sampling algorithms are either loss optimal or have infinite loss?
- Robustfill
- DeepCoder
- Etc.
- Paved the way in a coubple of different ways, etc.

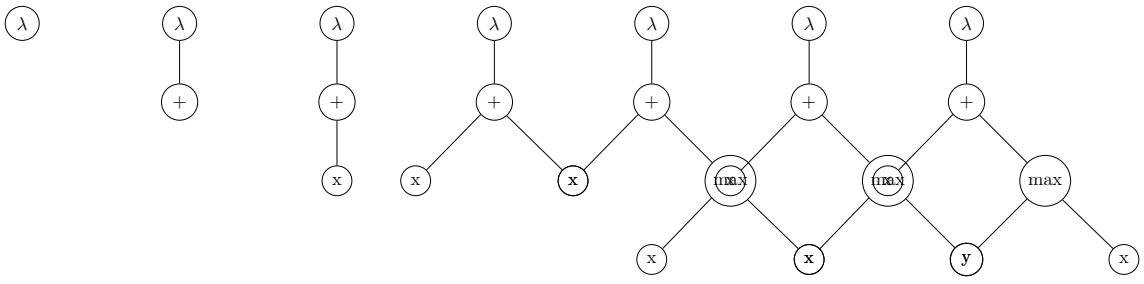

Figure 10: Creation of trees showing the construction process.

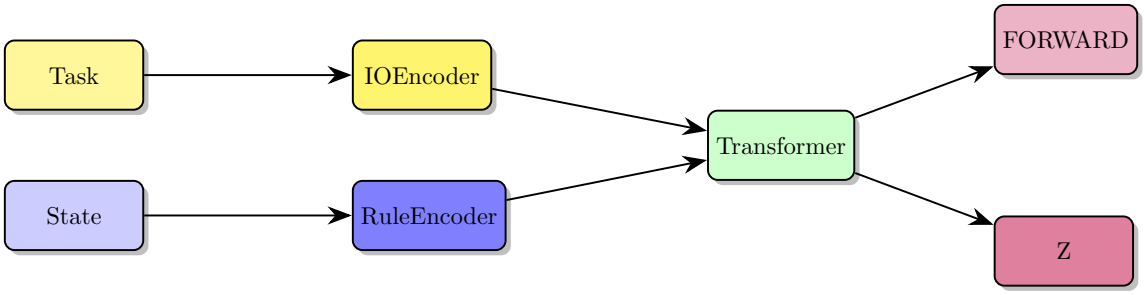

Figure 11: Insert explanation + maybe change task to an actual task; same with state; maybe also show the forward and Z output better.

Chapter 1

Philosophical Ramifications

1 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?

1.1 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 [23, 24].

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

1.2 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 computational boundary of the self

1.2.1 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"

1.2.2 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.

1.2.3 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.

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

[4] 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)."

- GLOM also related to NCA and to holarchies

- Kissner
- JEPA
- etc.

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?
- 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.

2 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]

3 What is Truth?

- 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.

- 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.

4 What is a Concept?

4.1 Identity and Essence

How do we discern between things?

Here we discuss homeostasis, allostasis — me against the world

4.1.1 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.)

4.1.2 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 [25]. 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.

4.2 concepts

if my idea of creating concepts is guided by discernment which may be a similar process to allostasis, homeostasis, how do I explain it with LLMs? Well LLMs just get all the data they need. In my idea we make discernments as we need. Thats how we build an ontology. First we may recognize the concept Animal, but then there is an animal which can fly and one that cannot. We call the ones that can fly birds. we then distinguish further.

A successful theory of concepts must be able to explain certain phenomena such as categorization, inference, and language use. Moreover, it has to be neurally implementable and support recursive binding. This means that concepts should be able to be composed and combined into new concepts. An adequate theory must explain why a given concept refers to some things and not others (in other words explain its extension), and it must also explain why this concept describes these referents in some ways and not others (in other words explain its intension). Blouw et al. [2] point out the wide scope such a theory must have; concepts can be:

- Perceivable objects (e.g. a table)

- Abstractions (e.g. love or money)
- Theoretical posits (e.g. the gene or the atom)
- Mathematical terms (e.g. addition, multiplication)
- Non-existent entities (e.g. a unicorn)
- And more
- is a concept a symbol?
- 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?

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?

- the transcendentals as the axioms of ontology
- proto-concepts
- concepts vs no concepts
- concepts as pointers
- normalizing over sensory data -¿ features -¿ objects -¿ concepts
- Concepts could be "chunked" into hyperdimensional vectors

4.3 The Hierarchy of Mental Representations: From Sensory Neurons to Conceptual Structures

- 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)

- relationship path
- mutual information (how well do concepts predict each other, co-occurence in reality)
- change distance
- parameter distance (related to hofstadter mathemagical themas) (how many knobs (or what is the minimal amount (related to edit distance or kologorov complexity)) would you have to twist to get from a a dog to a cat; from rock to jazz, ...)

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.

4.4 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 [26]. 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.

4.5 Concept space

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

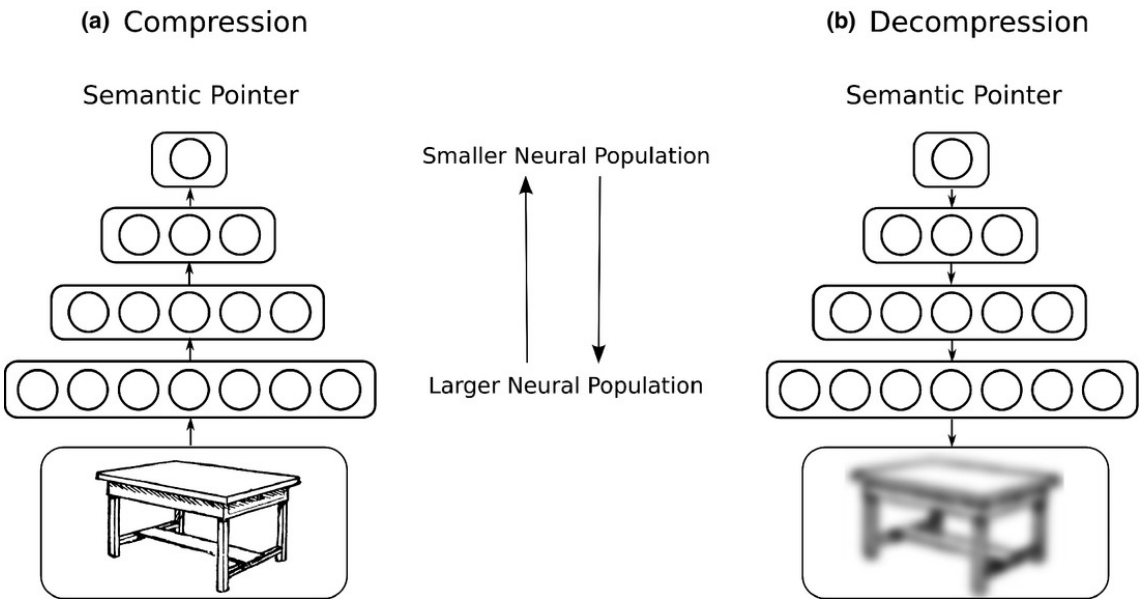

Figure 1.1: (a) A sensorimotor input, here a table, is compressed sequentially into a Semantic Pointer. (b) A Semantic Pointer is decompressed and returns a representation of a table. Due to the compression process the decompression results in noise. The figure is taken from Blouw et al. [2].

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 [27]. 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)

- Hyberbolic space (relate to holarchies)

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]

5 What is Meaning?

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

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 -

6 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?

7 What Does It Mean to Understand?

7.0.1 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, 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.

- are human's capabilities (or LLMs) just a bunch of bag of tricks, masquerading in a trench-coat.
- 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]

7.1 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.

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

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. Think of 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 What Is a Language?

- Children use language without knowing what the words actually mean. The meaning is the context.
- People can even teach without understanding. (so there seems to be a depth to understanding (maybe relate to SPA, shallow vs deep))
- Fake it 'til you make it?
- Is the limit of imitation a clone? ()

8.1 Language in Thought

- Syntax & Semantix
- Jabberwocky sentences (would be equivalent to a sentence that doesn't minimize surprise well (given the task, because it could also be a goal))
- Sapir-Whorf
- Why do we need language
- Does meaning emerge in LLMs (is meaning the network or do we need grounding)

- Grounding
- Binding problem (how do we assign meaning to geometrical features of stimuli)
- McLuhan.

Think about Wittgenstein and the notion that language is the limit of your reality. "The limits of my language mean the limits of my world." "the world we see is defined and given meaning by the words we choose. In short, the world is what we make of it."

Alfred Korzybski: our reality is limited by the human nervous system and our language Since we are limited by our nervous system and our language, we never have direct access to reality. "A map is not the territory it represents, but, if correct, it has a *similar structure* to the territory, which accounts for its usefulness."

This may relate to donald hoffman in that we have some handle on reality but it is obscured by our interface to it.

- Natural language as semiotic pointers to concepts. (think about alan moore to spell, etc. also advertisement, doublespeak, rhetoric, etc.)
- personality changes with language (hebrew vs german)
- English is an expressive language, a front-end, a communicative tool. Underneath, is a language of thought. It writes our conceptual world model. We are guided by our world-model, our ideology, our interpretation of reality and the truth and conclusions we derive from it.

  We can speak from compassion, from hate, from jealousy, apathy, love, and so on. This is not just emotional expression but it encodes our identity, attitude, and our beliefs.

  Language is symbolic, it point to concepts, and it is in the same class of concepts as money, a collective imaginary category.

  In order for our communication to be effective and useful, we need to create some shared framework. If we use different currencies and cannot convert between them, we cannot be in the same market. There needs to be some bridge that is accepted by both sides, so that we can interact. This is the same in language. We need to be able to translate. A translation is a mapping between two symbolic systems that point to the same concept.

  If we use the same language, but change its meaning, i.e. the symbolic pointers are orthogonal to each other, we cannot communicate anymore.

  A different way to think about this, is that symbolic systems, such as language, are relative and only work if we remain within the borders of the same context. Without a common frame-of-reference, language becomes meaningless. The pointers break.

-

8.1.1 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 [28]. 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: $\forall x A(x) \rightarrow (\exists y B(y) \wedge \neg \exists z C(z))$.

8.1.2 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)

9 What Is a Computation?

9.1 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.

- Talk about induction, deduction, abduction and how sampling from a reward distribution is similar to abduction aka inference to the best explanation

- add markpoels 7 criteria

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

[29] 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.

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.

9.2 Self-organization and predictive processing

[11] 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.

- 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

1. I think having a sense of self, i.e. a self-referential story, involves this inference model that builds causal structure, a narrative (see [30]).

2. Is this related to interpolation vs extrapolation? (is it the same in high dimensions? chollet? lecun?)

this subsection has to be unpacked.

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 [31]. 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 [32]. Fodor's theory implies an innate structure underlying human cognition, with systematic, rule-governed operations that manipulate symbols.

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 [33]. Turing showed that any possible computation can be done by a Turing machine and concluded that the human mind must be computational [34]. [Piccinini disagrees] Chomsky's Hierarchy classifies formal languages of different strengths and the automata that recognize them [35]. These results and many more are consequential when thinking about a language of thought. Many important questions about concepts are still left unanswered, e.g. regarding their formation, the structure of the framework they reside in, their representation, how they connect to other concepts and so on. 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 [12]. The idea is that we model the world by creating programs that generate our observations [36]. 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) [37]. Dehaene et al. further test this hypothesis and find similar results [12]. The question then remains, where primitives actually come from. 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 [38]. 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 [39]. 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 [40]. 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 [41].

Ellis et al. use λ-calculus, which is Turing complete, to build something akin to a LOT [1]. 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 [41, 42]. Moreover, purely symbolic attempts, albeit being the

original approach of artificial intelligence, have proven to be insufficient [28]. 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* [28].

- Relation to self-organization, active inference, FEP, autopoiesis, etc.
- what is a symbol and what is a symbolic architecture. symbolic here literally means symbol manipulation. context free grammar, etc. logical predicates. DC makes it neurosymbolic by using nns. I make it even more connectionist by embedding the symbols.
- 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.)

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

9.2.1 Poverty of the Stimulus

- arg for LOT

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.

Glossary

cognitive light cone . 37

Perception-Action Cycle . 39

Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

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

[11] 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.

[12] 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.

[13] 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].

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

[15] 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.

[16] 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.

[17] 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].

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

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

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

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

[29] 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.

[30] Nabil Bouizegarene, Maxwell Ramstead, Axel Constant, Karl Friston, and Laurence Kirmayer. Narrative as active inference, July 2020.

[31] 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.

[32] 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.

[33] 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.

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

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

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

[37] 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.

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

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

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

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

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

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 Parameters

Class	Parameter	Value
IOEncoder		
	size_max	10
	d_model	512
RuleEncoder		
	d_model	512
GFlowNet		
	d_model	512
	num_heads	8
	num_layers	2
	dropout	0.1
Training		
	min_program_depth	3
	max_program_depth	6
	epochs	145
	batch_size	4
	learning_rate_trn	1×10^{-4}
	learning_rate_gfn	1×10^{-4}
	e_steps	500
	m_step_threshold_init	150
	m_steps	150
	alpha	0.3
	beta	0.7
	epsilon	0.3
	replay_prob	0.3
	fantasy_prob	0.005

Table 1: Parameters

2.1 Big experiment

Table 2: Experiment Parameters

Variable	**Value**
d_model	512
max_program_depth	4
shuffle_tasks	True
n_tasks	145
variable_batch	False
train_ratio	0.5
seed	3
n_examples_max	(based on data.nb_examples_max)
size_max	10
lexicon	(based on data.lexicon)
cfg	(based on data.cfg)
num_heads	8
num_layers	2
dropout	0.1
min_program_depth	(based on data.max_program_depth)
max_program_depth	(based on data.max_program_depth)
epochs	5
batch_size	4
learning_rate_gen	1×10^{-4}
learning_rate_pol	1×10^{-4}
e_steps	2000
m_step_threshold_init	150
m_steps	2000
inference_steps	100
alpha	0.3
beta	0.7
epsilon	0.3
replay_prob	1
fantasy_prob	1
data	(based on data)
model	(based on model)
save_checkpoint	(based on save_checkpoint)