What makes Models Compositional? A Theoretical View

Parikshit Ram[1] , **Tim Klinger**[1] and **Alexander G. Gray**[2,3]
[1]IBM Research
[2]Centaur AI Institute
[3]Purdue University
parikshit.ram@ibm.com, tklinger@us.ibm.com, skirmilitor@gmail.com

Abstract

Compositionality is thought to be a key component of language, and various compositional benchmarks have been developed to empirically probe the compositional generalization of existing sequence processing models. These benchmarks often highlight failures of existing models, but it is not clear why these models fail in this way. In this paper, we seek to theoretically understand the role the compositional structure of the models plays in these failures and how this structure relates to their expressivity and sample complexity. We propose a general neuro-symbolic definition of compositional functions and their compositional complexity. We then show how various existing general and special purpose sequence processing models (such as recurrent, convolution and attention-based ones) fit this definition and use it to analyze their compositional complexity. Finally, we provide theoretical guarantees for the expressivity and systematic generalization of compositional models that explicitly depend on our proposed definition and highlighting factors which drive poor empirical performance.

1 Introduction

Compositionality is assumed to be integral to language processing [Pagin and Westerståhl, 2010a; Pagin and Westerståhl, 2010b]. Generalizing in a compositional manner or *compositional generalization* is of high interest when learning with sequences since it can enable a (learned) model to generalize well to a possibly infinite domain of sequences while learning from only a small number of examples. With this motivation, there has been interest in quantifying the compositional generalization of sequence or language models. This has led to various language modeling benchmarks such as SCAN [Lake and Baroni, 2018], CFQ [Keysers *et al.*, 2019], COGS [Kim and Linzen, 2020] and others [Andreas, 2018; Hupkes *et al.*, 2020], although compositional generalization can also be of interest in vision [Klinger *et al.*, 2020].

These benchmarks empirically probe the compositionality of off-the-shelf language models, often demonstrating the lack of compositional generalization. These highlight examples on which the models fail, but there is no precise understanding of why the failures occur. These findings are further put into question by results highlighting how such models can in fact compositionally generalize [Csordás *et al.*, 2021]. Nonetheless, various novel methods with improved compositional generalization (as measured by these benchmarks) have been developed [Russin *et al.*, 2019; Gordon *et al.*, 2019; Li *et al.*, 2019; Liu *et al.*, 2020; Nye *et al.*, 2020; Liu *et al.*, 2021], utilizing specialized models with compositional inductive biases. However, the area of compositional generalization is still *lacks a mathematical definition and measure of compositionality*.

Our contributions. Inspired by existing discussions on compositionality, and recent solutions for compositional generalization benchmarks, we make the following contributions: [1]

- We propose a general modular definition of "compositional functions" to facilitate concrete understanding of the expressiveness and generalization of such functions, and propose the notion of "compositional complexity" to quantify the complexity of such functions.
- We demonstrate the flexibility of this definition by highlighting how various existing models fit this definition, and how complex their compositions are.
- Given these definitions of compositional functions and compositional complexity, we precisely characterize the expressiveness and systematic generalization of such functions.

2 Related Work

A definition of compositionality [Pagin and Westerståhl, 2010a] states that the meaning $\mu(\cdot)$ of an expression is:

$$\mu(\alpha(u_1, \ldots, u_k)) = r_\alpha(\mu(u_1), \ldots, \mu(u_k)), \qquad (1)$$

where α is a (grammar) rule applied to the sub-terms u_i to obtain the expression $\alpha(u_1, \ldots, u_k)$, and r_α is a meaning operation that depends on the rule α. A non-technical phrasing of the principle of compositionality [Partee, 1995] is(quoted from Hupkes *et al.* [2020]): *"The meaning of a whole is a function of the meanings of the parts and of the way they are syntactically combined."* Among the expected properties of compositional functions are **systematicity** – the ability to

[1]A preliminary version of this work [Ram *et al.*, 2023] was previously presented at the KBCG@IJCAI23 workshop, while the supplementary material for this submission [Ram *et al.*, 2024] can be found at https://www.arxiv.org/abs/2405.02350.

consistently handle unknown combinations of known parts, and **productivity** – the ability to *handle arbitrary length sequences*. For systematic generalization, a model learned from some examples ("known parts") is expected to generalize to unseen examples ("unknown combinations"). Productive generalization requires learned models to generalize to sequences longer than those seen during learning.

Understanding compositionality and its relation to systematicity has been of long interest, and Zadrozny [1994] showed that compositional semantics can be defined for any meaning function without necessarily inducing systematicity. Recently, Rosenbaum *et al.* [2019] studied how networks (specifically routing networks) can be composed of multiple modules, focusing mainly on the training dynamics and empirical performance of such compositional models, while Wiedemer *et al.* [2023] try to formalize compositional generalization from the perspective of disentangled representations. In contrast to these works, we focus on understanding the compositionality of functions in terms of hierarchical computation needed to process the input (sequences), and the kinds of hierarchy induced by existing sequence processing models, such as recurrent, convolutional and attention-based models.

Dziri *et al.* [2023] empirically probe the compositionality of transformers on problems that require *"strict multi-step computations to derive correct answers"*, and consider the hierarchical computation necessary for the problem solving similar to our definitions. While Dziri *et al.* [2023] and others [Kim and Linzen, 2020; Sikarwar *et al.*, 2022; Ontanon *et al.*, 2022] show that transformers struggle with such tasks, a separate line of work highlight how transformers can be successful on compositional benchmarks [Csordás *et al.*, 2021; Zhou *et al.*, 2023; Drozdov *et al.*, 2023], which begs the questions (i) *whether* models (such as transformers) are *at all capable* of solving compositional tasks, and (ii) *when* such models are *able* to solve compositional tasks. Our proposed framework allows us to study such questions theoretically.

Jarvis *et al.* [2023] study systematicity for both datasets and functions, showing that modular functions can systematically generalize on datasets with compositional sub-structures, while general purpose non-modular functions do not. However, if the function modularity is unable to appropriately segregate the compositional sub-structure, even modular functions fail. To that end, they study a space of datasets with cleanly separable systematic sub-structures. In the context of Eq. (1), cleanly-separable systematic sub-structures correspond to non-overlapping sub-terms $u_i, i \in [\![k]\!]$. Our proposed framework can be seen as a generalization where the compositional sub-structures may not always be cleanly separable, but still present, and we study general purpose sequence models and their ability to express such compositional sub-structures.

3 Defining Compositionality

Our goal is to ground this principle into a mathematical form that will allow us to *quantify* the compositionality of models and understand how this quantification affects downstream compositional generalization. We define compositional functions $f : \mathcal{X} \to \mathcal{Y}$ with the domain $\mathcal{X}$ of input sequences $X = \{x_1, \ldots, x_L\}$ of atoms or tokens $x_i \in \mathcal{I}$ from an input dictionary $\mathcal{I}$. The range $\mathcal{Y}$ of f can be $\mathbb{R}$ for regression, $\{0, 1\}$ for binary classification, or $\mathcal{I}$ for next token prediction.

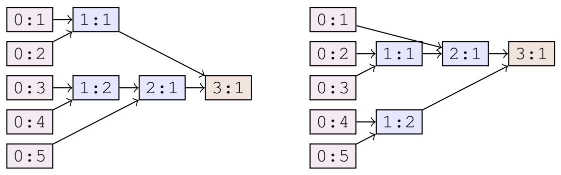


Figure 1: **cDAG**s for $f(X)$ (left) and $f(X')$ (right) in Example 1. Nodes are labeled $l{:}i$ (level l, index i). Sources are Fuchsia, sinks are Sepia, and internal nodes are Blue.

Definition 1. *To define f, we need the following components:*

- *Token encoder $e : \mathcal{I} \times \mathbb{N} \to \mathcal{H}$ (latent space), with $e_i = e(x_i, i) \in \mathcal{H}$ encoding the i^{th} token in $X \in \mathcal{X}$.*
- *A computation directed acyclic graph (DAG) or **cDAG** $D : \mathcal{X} \to \mathcal{D}$ (the space of DAGs), with $D(X)$ defining the hierarchical processing of a sequence X. $D(X)$ can also be viewed as the trace of program used by function f to process X. We will describe this in further detail soon.*
- *Span processor $g : \mathcal{H}^k \to \mathcal{H}$ maps k terms in the latent space into a new term in the latent space.*
- *Read-out function $h : \mathcal{H}^m \to \mathcal{Y}$ which maps the final set of terms in the latent space to the output space $\mathcal{Y}$.*

Given the above, we define a compositional function as

$$f(X) = h\left(g^{\otimes D(X)}(e(x_1, 1), \ldots, e(x_L, L))\right), \qquad (2)$$

where $g^{\otimes D(X)}$ is the recursive operation of g over $D(X)$.

Next we further discuss the components.

A **computation DAG** or **cDAG** $D(X) \triangleq \{N(X), E(X)\}$ for a specific input sequence $X \in \mathcal{X}$ can depend on X or be pre-specified. This **cDAG** is a leveled DAG with set of nodes $N(X)$ and edges $E(X)$. Each node $n \triangleq (l : i) \in N(X)$ has a level l and index i (see Appendix A.1 in supplement for details). The recursive application of g over $D(X)$ induces a value $v_{l:i} \in \mathcal{H}$ for each internal node $n \in N(X)$. The sources is $N(X)$ have level 0, and there is one source for each $x_i \in X, i \in [\![L]\!] \triangleq \{1, \ldots, L\}$ with index i and value $v_{0:i} = e(x_i, i) \in \mathcal{H}$. There are m sinks in $N(X)$, and at most k incoming edges and q outgoing edges at any node. For an internal node $n \in N(X)$ with k parents $P(n)$, the value $v_{l:i} = g(v_{l_1:i_1}, \ldots, v_{l_k:i_k}) \in \mathcal{H}$ where $v_{l_j:i_j}$ is the value of the j^{th} parent in $P(n)$. Note that this **cDAG** corresponds to the "forward-pass" for inference.

We consider the explicit **cDAG** because it allows us to see how the different elements $x_i, i \in [\![L]\!]$ of the input sequence X are hierarchically composed to obtain the output. This will allow us to study the complexity of any compositional function. A "simple" **cDAG**, where all source nodes just connect to a single sink node, would be "applicable" to all functions, but it does not allow us to study it in an interesting manner. When we study the compositional functions induced by general purpose models (such as recurrent, convolutional or transformer models), we will see that some models have explicit **cDAG**s with more structure, while others have less structured explicit **cDAG**s, but there are implicit structures induced in the **cDAG**; whenever possible, we will explicitly state this implicit structure and study its properties. From a neuro-symbolic perspective [Sarker *et al.*, 2021; Garcez *et al.*, 2022], this explicit **cDAG** can be seen as the symbolic part, while the e, g, h are the neural; note that, in some models, this symbolic **cDAG** might be created with neural elements, while in others, the **cDAG** might be obtained with a symbolic grammar. This neuro-symbolic view offers a novel theoretical understanding of compositionality.

The **span processor** $g : \mathcal{H}^k \to \mathcal{H}$ takes as input k elements from the latent space $\mathcal{H}$ and outputs an element in $\mathcal{H}$. While the definition implies that the same g needs to be operated recursively over the **cDAG** $D(X)$, there is no restriction on the inputs and output of g regarding the information encoded in the latent space. For example, if the level l of any node $l{:}i$ is encoded into its value $v_{l:i}$, then the g will behave differently across levels (*level-dependent*); if the index i of the node $l{:}i$ is encoded into its value, then g will be sensitive to the positional information (*order-dependent*); if the value of a node includes the type of the node (for example, a non-terminal in a grammar), then g can be *type-dependent*. Our definition states that the arity of the span processor $g : \mathcal{H}^k \to \mathcal{H}$ is k. We do so for the ease of exposition, though our definition can incorporate more flexible span processors (see Appendix A.2 in supplement).

The **read-out function** $h : \mathcal{H}^m \to \mathcal{Y}$ finally maps m elements in the latent space to the output space $\mathcal{Y}$. This separation between g and h was necessary in our proposed definition because we require g to be operable recursively, and thus g can operate in a latent space $\mathcal{H}$ distinct from $\mathcal{Y}$. In some applications, $\mathcal{H} \supseteq \mathcal{Y}$, in which case, h can be an identity function. In an alternate scenario, where the g function is identity, and the **cDAG** function produces "trivial **cDAG**s" – **cDAG**s where the source nodes are the sink nodes (see Fig. 3a for example), h would effectively be a mapping from $\mathcal{X} \to \mathcal{Y}$ (subsuming the token encoder within h). But in this scenario, we are not able to explicitly view the recursive operation desired in the "*meaning of the whole is a function of the meaning of the parts*" principle of compositionality. Hence, we make this separation between h and g explicit. There are couple of aspects of this read-out function we wish to discuss explicitly – (i) We assume that h is specifically *non-compositional* and processes its input without breaking it up into any sub-problems; we explicitly define the compositional function f separating out g, D, h, where g (neural) and D (symbolic) represent the compositional part. (ii) We require h to have a fixed-arity of m since g and D are aggregating the information over the input.

Example 1. *Figure 1 (left) shows the* **cDAG** $D(X)$ *for a compositional* f *on* $X = [x_1, \dots, x_5]$, *with* $f(X) = h\left(g\left(g\left(e_1, e_2\right), g\left(g(e_3, e_4), e_5\right)\right)\right)$, $k = 2$ *in-degree,* $q = 1$ *out-degree,* $m = 1$ *sink,* $e_i = e(x_i, i) \in \mathcal{H}$, *span-processor* $g : \mathcal{H}^2 \to \mathcal{H}$, *and read-out function* $h : \mathcal{H} \to \mathcal{Y}$. *The values* $v_{0:i} = e_i$ *for sources* $0{:}i$, $i \in \{1, \dots, 5\}$, *and the internal node values are:* $v_{1:1} \leftarrow g(e_1, e_2)$, $v_{1:2} \leftarrow g(e_3, e_4)$, $v_{2:1} \leftarrow g(v_{1:2}, e_5)$, $v_{3:1} \leftarrow g(v_{1:1}, v_{2:1})$. h *operates on* $v_{3:1}$ *at sink* $3{:}1$. *Figure 1 (right) shows the* **cDAG** $D(X')$ *of the same* f *on* $X' \neq X$ *with the same* $k = 2, q = 1, m = 1$ *and* $f(X') = h\left(g(g(e_1, g(e_2, e_3)), g(e_4, e_5))\right)$.

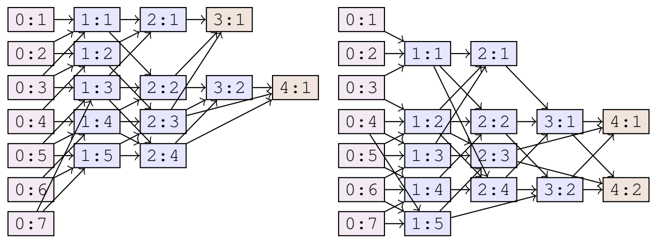


Figure 2: **cDAG**s for $\mathsf{f}(X)$ (left) and $\mathsf{f}(X')$ (right) in Example 2.

Example 2. *Figure 2 (left) shows the* **cDAG** $\mathsf{D}(X)$ *for a compositional* f *on* $X = [x_1, \dots, x_7]$, *with* $\mathsf{f}(X) = \mathsf{h}\left(v_{4:1}, v_{3:1}\right)$, $k = 3$ *maximum in-degree,* $q = 3$ *maximum out-degree,* $m = 2$ *sinks,* $e_i = e(x_i, i) \in \mathcal{H}$, *span processor* $\mathsf{g} : \mathcal{H}^3 \to \mathcal{H}$, *and read-out function* $\mathsf{h} : \mathcal{H}^2 \to \mathcal{Y}$. *The source values* $v_{0:i} = e_i$ *for each* $i \in \{1, \dots, 7\}$, *and the internal node values are:* $v_{1:1} \leftarrow \mathsf{g}(e_1, e_2, e_3)$, $v_{1:2} \leftarrow \mathsf{g}(e_2, e_3, e_4)$, $v_{1:3} \leftarrow \mathsf{g}(e_3, e_5, e_7)$, $v_{1:4} \leftarrow \mathsf{g}(e_4, e_5, e_6)$, $v_{1:5} \leftarrow \mathsf{g}(e_5, e_6, e_7)$, $v_{2:1} \leftarrow \mathsf{g}(v_{1:1}, v_{1:2}, v_{1:3})$, $v_{2:2} \leftarrow \mathsf{g}(v_{1:1}, v_{1:3}, v_{1:4})$, $v_{2:3} \leftarrow \mathsf{g}(v_{1:2}, v_{1:4}, v_{1:5})$, $v_{2:4} \leftarrow \mathsf{g}(v_{1:3}, v_{1:4}, v_{1:5})$, $v_{3:1} \leftarrow \mathsf{g}(v_{2:1}, v_{2:2}, v_{2:3})$, $v_{3:2} \leftarrow \mathsf{g}(v_{2:2}, v_{2:3}, v_{2:4})$, $v_{4:1} \leftarrow \mathsf{g}(v_{3:2}, v_{2:3}, v_{2:4})$. h *operates on* $v_{3:1}$ *and* $v_{4:1}$ *at sinks* $3{:}1$ *and* $4{:}1$. *Figure 2 (right) shows the* **cDAG** $\mathsf{D}(X')$ *of the same* f *on* $X' \neq X$ *with the same* $k = 3, q = 3, m = 2$.

While Example 1 is a simple compositional function on a sequence, Example 2 is a more sophisticated one. This is to highlight that our proposed Definition 1 can handle functions which require more complex interactions between the tokens in a sequence. Example 1 has a **cDAG** with a maximum out-degree $q = 1$, implying a single path from any source to a sink. Example 2 has a **cDAG** with a maximum out-degree $q = 3$ across all levels in the DAG, implying that there can be a large number of paths to any sink from a source. This allows the definition to include functions where certain tokens in the sequence are of much higher importance to the output than others. These examples also highlight that edges in the **cDAG** are allowed to skip levels, and the sinks can be from different levels, further highlighting the compositional flexibility.

We like to remark on a couple of points here: (i) Through these examples, we show that our definition explicitly considers how the problem of sequence processing is broken up into sub-problems – the **cDAG** embodies how disjoint or intertwined these "sub-problems" are by explicitly considering the computation hierarchy. (ii) For input sequences X, X' from the same problem domain, and the same compositional function f, we allow the **cDAG** to be different – **cDAG** $D(X)$ can be input-dependent – thereby allowing different input sequences to have different sub-problem hierarchies.

Before we discuss properties of this form of compositional functions, we note that *such a precise yet flexible definition is one of our contributions*, and we will show how existing models (architectures) fit this definition. It is a precise elaboration of the succinct recursive Eq. (1) – we make precise the recursion, and how the sub-terms u_i are recursively built up. At a non-technical level, we also believe that our proposed

Definition 1 connects intuitively to existing definitions:

$$\underbrace{\text{The meaning of the whole}}_{f:\mathcal{X}\to\mathcal{Y}} \text{ is a } \underbrace{\text{function}}_{h:\mathcal{H}^m\to\mathcal{Y}} \text{ of } \underbrace{\text{the meanings of the parts}}_{g:\mathcal{H}^k\to\mathcal{H}}$$
$$\text{and } \underbrace{\text{of the way they are syntactically combined.}}_{D:\mathcal{X}\to\mathcal{D}}$$

Both Examples 1 and 2 can be seen as compositional functions, but Example 2 is clearly a more complex composition. In addition to its intuitive nature, our proposed definition allows us to understand *how complex the compositionality is* beyond just stating if a function is compositional.

Compositional Complexity. This depends on the functions g, h, e as well as the **cDAG** function D that drives the computation. For a sequence X of length L, $D(X)$ has L source nodes, maximum in-degree of k (controlling the span size for g), m sink nodes (controlling the capacity of h), maximum out-degree of q (quantifying the "localism" of the effect of a node). However, these do not explicitly incorporate the fact that changes to nodes at lower levels of the **cDAG** *can* have a larger effect on the output than changes to nodes at higher levels of the **cDAG**. Instead, we propose a new quantification – the *locus of influence* or LoI of any source node:

Definition 2 (LoI of a source node)**.** *Consider a compositional function f with components e, D, g, h (as in Definition 1). Let $(v_{n_1}, \ldots, v_{n_j}, \ldots, v_{n_k}) \in \mathcal{H}^k$ be any input to the span processor g, with $v_n = g(v_{n_1}, \ldots, v_{n_j}, \ldots, v_{n_k})$ its output. Let $\varepsilon \in \mathcal{H}$ be a "perturbation" to the j^{th} argument to g, $j \in [\![k]\!]$, resulting in the perturbed output $v_n^j(\varepsilon) = g(v_{n_1}, \ldots, v_{n_j} + \varepsilon, \ldots, v_{n_k})$. Let $c > 0$ be an universal constant such that $\forall j \in [\![k]\!]$, $\forall \varepsilon \in \mathcal{H}$,*

$$\left\|v_n - v_n^j(\varepsilon)\right\| \leq c\|\varepsilon\|. \tag{3}$$

For a sequence $X \in \mathcal{X}$ of length L, and a source node $0:i$ in $D(X)$, let $P(x_i)$ be the set of all unique paths from $0:i$ to any of the sink nodes in $D(X)$. We define the absolute LoI of index i as $\delta_i = \sum_{P\in P(x_i)} c^{|P|}$, with $|P|$ as the length of a path $P \in P(x_i)$, and the relative LoI as $\beta_i = \delta_i / \sum_{j\in[\![L]\!]} \delta_j$.

This definition of the complexity of composition incorporates both the complexity of the **cDAG** $D(X)$ and the complexity of the span processor $g : \mathcal{H}^k \to \mathcal{H}$ in terms of its smoothness, with higher values of c indicating more complex (less smooth) g. The absolute LoI δ_i incorporates the effect of longer paths, with the effect growing with path length, and corresponds to the sensitivity of the compositional function output to any one input token in the sequence.

The smaller the absolute LoI δ_i of any input index i, more local its effect, and thus more structure that can be transferred between examples if x_i is replaced with something else. A relative LoI β_i greater than $1/L$ denotes that the input index i (and thus input token x_i) has an out-sized effect on $D(X)$ (and thus the computation) compared to the other indices (tokens). In Example 1 (left), $\delta_1 = c^2, \beta_1 = {}^1/_{2c+3} < {}^1/_5$ while $\delta_3 = c^3, \beta_3 = {}^c/_{2c+3} > {}^1/_5$, implying that x_3 has more influence (absolute and relative) function than x_1 (assuming $c > 1$). In Example 2 (left), $\delta_1 = c^4 + 2c^3, \beta_1 = {}^{c+2}/_{27c+39} \approx {}^1/_{22} < {}^1/_7$, while $\delta_5 = 7c^4 + 9c^3, \beta_5 = {}^{7c+9}/_{27c+39} \approx {}^1/_4 > {}^1/_7$, hence x_5 has a significantly larger influence than x_1.

We utilize the LoI to define the complexity of a compositional function, and a class of such compositional functions:

Definition 3. *A function $f : \mathcal{X} \to \mathcal{Y}$ with components g, h, e, D is $(k, q, m, \boldsymbol{\delta}, \boldsymbol{\beta})$-compositional if, for any $X \in \mathcal{X}$ of length L (that is, $|X| = L$), the **cDAG** $D(X)$ has a in-degree of k, maximum outgoing degree of q, and m sink nodes, and for $\forall i \in [\![L]\!], \delta_i \leq \boldsymbol{\delta}$, and $\beta_i \leq \boldsymbol{\beta} \in [1/L, 1)$. We denote with $\mathcal{F}$ a class of such $(k, q, m, \boldsymbol{\delta}, \boldsymbol{\beta})$-compositional functions.*

A small $\boldsymbol{\delta}$ and a $\boldsymbol{\beta}$ close to $1/L$ signifies a function that possesses a high level of localism across all input sequences and tokens in its domain. While this function has the most structure, it might not be suitable for practical purposes. A high $\boldsymbol{\delta}$ and a $\boldsymbol{\beta}$ close to $1/L$ signifies a very complex function where there is a lot of interaction between all the input tokens in all input sequences, making it hard to exploit any compositional structure in the function. A high $\boldsymbol{\delta}$ and a $\boldsymbol{\beta}$ significantly higher than $1/L$ indicates an interesting class of functions where, some input tokens *can have a high influence over the function computation*, but, for most tokens, there is a compositional structure in the function that can be exploited. This intuitively seems to be an interesting and more practical class of compositional functions since assuming all tokens have an equal level of relative influence seems quite restrictive.

Cleanly-separable systematic sub-structures. Revisiting Eq. (1), where u_i are sub-parts, we highlight a special case of our proposed Definition 3 of compositional functions: If the sub-parts $u_i, u_j, i \neq j$ are non-overlapping throughout the recursion, up to the base case where the sub-parts, u_i, are the tokens in the input, then it would induce a **cDAG** with a maximum outgoing degree $q = 1$. This implies that the number of paths $|P(x_i)| = 1$ for any source $0{:}i$ – there is a single source-to-sink path for any source, resulting in a **cDAG** with significantly reduced compositional complexity.

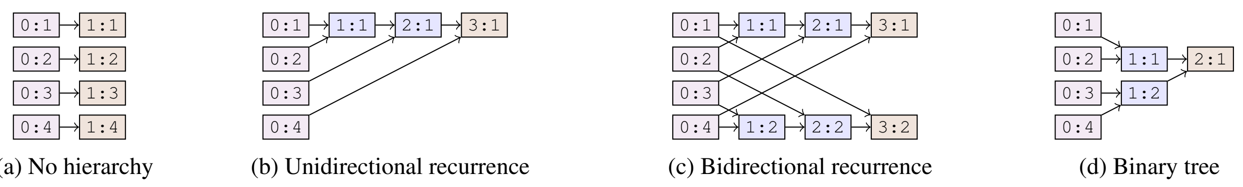
4 Existing Models as Compositional Functions

Here we will discuss various model classes induced by existing model architectures, how they fit Definition 1 of compositional functions, and how they compare to each other. We will re-express existing sequence processing models as per our definition, teasing out the symbolic **cDAG** (and the neural g, h) and studying their compositional complexity. The presented **cDAG** for each model class corresponds to a "forward-pass" for inference. Omitted technical details are in Appendix B in the supplement.

We begin with the trivial **cDAG** of **no hierarchical composition** in Fig. 3a. All sources in $D(X)$ connect to separate sinks. The composition is written as $h(g(e_1), g(e_2), g(e_3), g(e_4))$. For L-length inputs, this is a $(k = 1, q = 1, m = L, \boldsymbol{\delta} = c, \boldsymbol{\beta} = {}^1/_L)$-compositional function class.

4.1 Recurrent Composition

Figure 3b presents an example **cDAG** for **unidirectional recurrent composition**, with the corresponding compositional function $h(g(g(g(e_1, e_2), e_3), e_4))$. Specific choices of the g and h functions would give us specific recurrent neural network (RNN) models. The **cDAG** recursively combines 2 nodes in the order of the original sequence to get a class of compositional functions with $(k = 2, q = 1, m = 1)$. The **cDAG**



(a) No hierarchy (b) Unidirectional recurrence (c) Bidirectional recurrence (d) Binary tree

Figure 3: Existing models - I. Nodes are labeled l:i (level l, index i). Sources are Fuchsia, sinks are Sepia, and internal nodes are Blue.

is input-agnostic, that is $D(X) = D(X')\forall X, X' \in \mathcal{X}$ with $|X| = |X'|$. This composition can operate on arbitrary length sequences, and its complexity is quantified as:

Proposition 1. *With unidirectional recurrent composition, the maximum absolute LoI is* $\boldsymbol{\delta} \triangleq c^{L-1}$*, with a maximum relative LoI of* $\boldsymbol{\beta} \triangleq (c^L - c^{L-1})/(2c^L - c^{L-1} - 1)$.

The absolute LoI is large (exponential in the input length), and the relative LoI is close to $1/2 \gg 1/L$ for large enough L.

Figure 3c is an example of a **cDAG** of a **bidirectional recurrent composition**, which can be written algebraically as $h(g(g(g(e_1, e_2), e_3), e_4), g(g(g(e_4, e_3), e_2), e_1))$. The **cDAG** recursively combines 2 nodes in the order of the sequence, then in the reverse order, giving us a function class with $(k = 2, q = 2, m = 2)$. This is similar to the unidirectional recurrence, but with two sink nodes instead of one, and can operate on arbitrary length sequences but the **cDAG** is input-agnostic. The compositional complexity is given by:

Proposition 2. *With bidirectional recurrent composition, the maximum absolute LoI is* $\boldsymbol{\delta} \triangleq c^{L-1} + c$*, with a maximum relative LoI of* $\boldsymbol{\beta} \triangleq (c^L - c^{L-1} + c^2 - c)/2(2c^L - c^{L-1} - 1)$.

Note that, while the $\boldsymbol{\delta}$ for the bidirectional recurrent composition remains of the same order as that of the unidirectional recurrent one, $\boldsymbol{\beta}$ is approximately halved (approaching $1/4$).

Figure 3d shows an example of a **balanced tree recurrent composition** by utilizing a balanced binary-tree **cDAG** with $(k = 2, q = 1, m = 1)$ (as in a TreeLSTM [Tai *et al.*, 2015]), with an algebraic form $h(g(g(e_1, e_2), g(e_3, e_4)))$. The **cDAG** is input-agnostic (like the previous two), but can operate on arbitrary length sequences. The compositional complexity is:

Proposition 3. *With balanced binary-tree recurrent composition, the maximum absolute LoI is* $\boldsymbol{\delta} \triangleq c^{\lceil \log_2 L \rceil}$*, with a maximum relative LoI of* $\boldsymbol{\beta} \triangleq 1/L$.

This form of composition significantly reduces the complexity of the **cDAG** relative to the previous two models both in terms of $\boldsymbol{\delta}$ (linear dependence in L instead of exponential) and $\boldsymbol{\beta}$ ($1/L$ instead of a constant $\gg 1/L$). There are versions of the tree recurrent composition that leverage the parse tree of the input to define the **cDAG** [Socher *et al.*, 2010], and thus, natively fit our definition. Here the **cDAG** will no longer be input-agnostic, and Proposition 3 would not apply; the complexity will depend on the grammar driving the input-dependent parse trees. Various models [Bowman *et al.*, 2016; Shen *et al.*, 2018; Shen *et al.*, 2019] integrate parse-tree structures into a RNN for an input-dependent **cDAG** for the recurrent composition. They learn to generate a parse-tree for any given input (either via supervision from an external parser or directly from the data). All these models fit our definition of compositional functions with an input-dependent tree-based **cDAG**.

4.2 Convolutional Composition with Pooling

Figure 4a shows an example **cDAG** induced by repeated application of **convolution-then-pooling** or conv+pool, performing a convolution over a span of size 2, and pooling over a span of size 2 in this example (see Figure 5a in supplement for a detailed version of Fig. 4a that shows how repeated conv+pool implicitly induces the **cDAG** in Fig. 4a; Figure 5e and corresponding **cDAG** in Figure 5f in the supplement provides an example of convolution with padding over a span of size 2 and pooling over a span of 3). The composition in Fig. 4a can be algebraically written as $h(g(g(e_1, e_2, e_3), g(e_3, e_4, e_5), g(e_5, e_6, e_7)))$. In general, we can consider a span processor g function that performs convolution over a span of size w and then pools over a span of p. Here, the number of sinks m has to be user-specified, and the **cDAG** repeatedly applies conv+pool until it reduces the number of nodes at the highest level to m. This induces a compositional function class with $(k = w + p - 1, q = w)$.

Proposition 4. *Assuming that* $1 < w, p \ll L$*, the conv+pool composition has a maximum absolute LoI of* $\boldsymbol{\delta} \sim \mathcal{O}(c^{\log L})$*, and a maximum relative LoI of* $\boldsymbol{\beta} \sim \mathcal{O}(2/(L(1 + 1/p)))$.

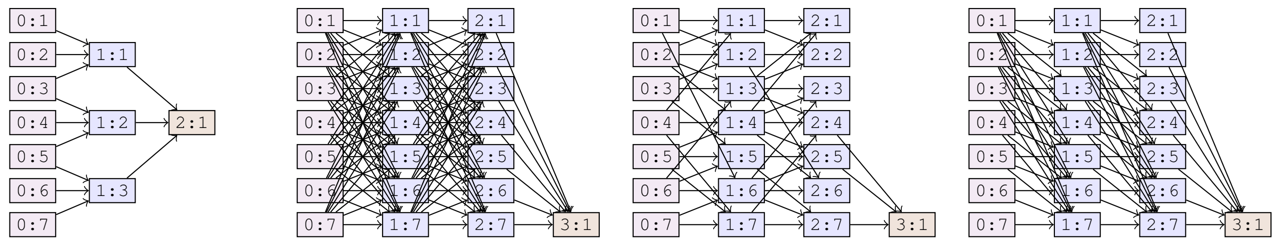
We would like to highlight an important subtlety here: If we utilize average or sum pooling, the **cDAG** is input-agnostic; if the pooling is selective, such as in max-pooling or min-pooling, *some of the edges in the* ***cDAG*** *implicitly get deactivated during the recursive computation of the function, leading to an input-dependent* ***cDAG*** – the **cDAG** itself depends on the input sequence, and can change between inputs of same length. Figure 5c shows an example of selective pooling, with the corresponding input-dependent **cDAG** shown in Figure 5d. This function can operate on arbitrary length input sequences.

4.3 Transformers

Figure 4b expresses the **cDAG** of a **transformer** with M levels or "blocks", where the span-processor $g : \mathcal{H}^k \to \mathcal{H}$ is a transformer block with the (multi-headed) self-attention, residual connections, layer normalization, and the token-wise ReLU network [Vaswani *et al.*, 2017] (see Appendix B.5 in the supplement). We consider the version where, for inference, only the last token representation (after M transformer blocks) is utilized for prediction – the read-out function only applies to the last token representation. For a input sequence of length L, $k = q = L$, with $m = 1$ sink node. The compositional complexity is:

Proposition 5. *A transformer based composition with* M *blocks has a the maximum absolute LoI of* $\boldsymbol{\delta} = L^{M+1}c^{M+1}$*, and a maximum relative LoI of* $\boldsymbol{\beta} = 1/L$.

This function class has a really high $\boldsymbol{\delta}$ (exponential in the number of blocks M but polynomial in the input length L



(a) Convolve + Pool (b) Transformer (c) Transformer + sparse att. (d) Decoder-only Transformer

Figure 4: Existing models - II. Nodes are labeled $l{:}i$ (level l, index i). Sources are Fuchsia, sinks are Sepia, and internal nodes are Blue.

unless $M > L$) and the lowest possible $\boldsymbol{\beta} = 1/L$ – the output is equally sensitive to all tokens in the sequence. It is interesting to contrast this complexity to that of a balanced binary-tree recurrent composition, which has the same level of relative LoI $\boldsymbol{\beta} = 1/L$, but the $\boldsymbol{\delta}$ is linear in L instead of polynomial (or exponential if $M > L$); the unidirectional and bidirectional recurrent compositions have an exponential dependence on the length L (but much larger $\boldsymbol{\beta}$).

This composition does not explicitly have an input-dependent **cDAG** – all nodes at a level always connect to all nodes at the next level since all attention (scores) are positive. In practice, the attention weights are obtained with softmax activation, which pushes the attention to be approximately sparse. If we consider the attention to be exactly sparse [Tay *et al.*, 2022] with $K \ll L$ nonzero scores (rest zero), the **cDAG** will have much fewer edges as in Fig. 4c (here, $K = 3$). If the attention weights and sparsity pattern are input-dependent (as in with top-K attention [Gupta *et al.*, 2021]), transformers with hard attention will have input-dependent **cDAG**s. Here $k = K, q = L, m = 1$, and the compositional complexity is:

Proposition 6. *A transformer with M blocks and K-sparse attention ($K \ll L$) has a maximum absolute LoI of $\boldsymbol{\delta} = LK^M c^{M+1}$, and a maximum relative LoI of $\boldsymbol{\beta} = 1/K$.*

Contrasting this to the complexity of the standard transformer (Proposition 5), we see that the $\boldsymbol{\delta}$ is considerably lower ($\mathcal{O}(LL^M)$ vs $\mathcal{O}(LK^M)$ with $K \ll L$), implying that the LoI of any one input token (thus the sensitivity) is substantially reduced. However, the relative LoI $\boldsymbol{\beta}$ can be considerably higher ($1/L$ vs $1/K$), indicating that the composition is allowed to be relatively more sensitive to certain tokens than others. Attention-based span-processor g can handle arbitrary length inputs (albeit at quadratic computation per block/level), and the read-out function h always has a fixed arity of $m = 1$. Thus, this transformer can operate on arbitrary length inputs.

If we consider a **decoder-only transformer**, then for all levels $l < M$, all edges from nodes $l{:}i \to l{+}1{:}j$ for $j < i$ would no longer be in the **cDAG** (Fig. 4d). The complexity is:

Proposition 7. *A decoder-only transformer with M blocks has a maximum absolute LoI is $\boldsymbol{\delta} = L^M c^{M+1}$, and a maximum relative LoI of $\boldsymbol{\beta} = 1/(1 + \sum_{i \in [\![L-1]\!]} ({}^{i}/_{L})^M)$.*

As L or M grows, the sum $\sum_i ({}^{i}/_{L})^M$ goes to zero, and thus, $\boldsymbol{\beta} \to 1$. For some $\Delta \in (0, 1)$, $\boldsymbol{\beta} \triangleq 1/(1 + r\Delta)$ for the largest $r \in [\![L]\!]$ such that $\Delta \leq (1 - r/L)^M$. We can similarly study the decoder-only transformer with sparse/hard attention.

Model	IC	AL	(k, q, m)	$\boldsymbol{\delta}$	$\boldsymbol{\beta}$
No-hierarchy (3a)	✗	✗	$(1, 1, L)$	c	$1/L$
Uni-RNN (3b)	✗	✓	$(2, 1, 1)$	c^{L-1}	$1/2$
Bi-RNN (3c)	✗	✓	$(2, 2, 2)$	c^{L-1}	$1/4$
Tree-RNN (3d)	⧫	✓	$(2, 1, 1)$	$c^{\log L}$	$1/L$
Conv+pool (4a)	†	‡	$(w{+}p, w, m)$	$c^{\log L}$	$\frac{2/L}{1+1/p}$
Transformer (4b)	✗	✓	$(L, L, 1)$	$(Lc)^M$	$1/L$
SpAtt Trf (4c)	✓	✓	$(K, L, 1)$	$L(Kc)^M$	$1/K$
Decoder Trf (4d)•	✗	✓	$(L, L, 1)$	$(Lc)^M$	$\frac{1}{(1+r\Delta)}$

Table 1: Complexities of existing models. LoI is specified approximately for the ease of exposition. **IC:** Input-dependent **cDAG**. **AL:** Arbitrary length operation. ⧫: The **cDAG** can be input-dependent if a input parse tree is available. †: Conv+Pool induces input-dependent **cDAG**s for max/min-pool, not for avg/sum-pool. ‡: The number of sinks m needs to be specified for conv+pool, and the model can handle arbitrary length if it can recursively conv+pool until the number of nodes is reduced to m. •: See discussion after Proposition 7.

Efficient transformers. While transformers are able to handle arbitrary length input, their computational cost scales quadratically in the input length L (compared to the linear cost of recurrent models). To mitigate this issue, various "efficient" transformers have been proposed [Tay *et al.*, 2022; Lin *et al.*, 2022]. Various sparse attention mechanisms have been studied, utilizing block-local, dilated, global or banded attention (see Lin et al.[2022, Figure 4]). These architectures reduce the quadratic cost often to almost linear. In terms of their compositional complexity, these architectures can easily be studied within our proposed framework, and will have significantly lower complexity compared to the vanilla transformer (smaller $\boldsymbol{\delta}$, similar $\boldsymbol{\beta}$). However, these architectures induce an input-agnostic sparsity pattern, thus leading to input-agnostic **cDAG**s. In contrast, sparse attention schemes with input-dependent sparsity patterns such as top-K attention [Gupta *et al.*, 2021] and sparse Sinkhorn attention [Tay *et al.*, 2020] induce input-dependent **cDAG**s.

4.4 Discussion

We discussed various existing sequence processing models in the context of our definition of compositional functions (Definitions 1 and 3), and quantify their corresponding complexities in the form of bounds on the absolute LoI $\boldsymbol{\delta}$ and relative LoI $\boldsymbol{\beta}$. We summarize these properties in Table 1 to compare different models. Beyond complexities, we also specify (i) whether they can operate on arbitrarily long sequences, and (ii) whether the **cDAG**s are input-dependent or input-agnostic for some fixed input length L. In Appendix B.9, we discuss a model's ability to process arbitrary lengths, and how it relates to parameter sharing across different **cDAG** levels. Here, we focus on the topic of input-agnostic vs input-dependent **cDAG**s.

As we summarize in Table 1, the only model with explicitly input-dependent **cDAG** is the sparse attention versions of the transformer (and the tree based recurrence if the tree is input-dependent). The conv+pool composition implicitly induces an input-dependent **cDAG** if the pooling operation is explicitly selective (such as using max-pooling or min-pooling).

Given (i) the success of the softmax attention mechanism in sequence processing tasks, (ii) the use of explicitly input-dependent **cDAG**s in efficient transformers such as Sinkhorn transformers, tree-based RNNs, and specialized models for compositional generalization, and (iii) the wide use of max/min-pooling instead of sum/average pooling (especially in vision tasks), we think it is important to rigorously understand the value of input-dependent **cDAG**s. First, it is fair to assume that most realistic problems involve input-dependent **cDAG**s. For example, if the input, and the corresponding output, are generated from a grammar, then the **cDAG** of any input X would be closely related to its parse-tree.

To this end, we specifically study the ability of the compositional function with an input-agnostic **cDAG** to approximate a compositional function with input-dependent **cDAG**s. We present a condensed version of the result for ease of exposition; see Appendix C in the supplement for details and proof.

Theorem 1 (condensed)**.** *Consider a $(k, q, m, \boldsymbol{\delta}, \boldsymbol{\beta})$-compositional function class $\mathcal{F}$, and input sequences $X \in \mathcal{X}$ of length L. Consider a ground-truth function $f \in \mathcal{F}$ with components e, D, g, h, and a compositional function $\mathsf{f} \in \mathcal{F}$ with components $e, \mathsf{D}, \mathsf{g}, \mathsf{h}$, with an input-agnostic **cDAG** such that $\mathsf{D}(X) = \mathsf{D}\,\forall X \in \mathcal{X}, |X| = L$.* [2] *Then the worst-case approximation of f by f, for $f, \mathsf{f} \in \mathcal{F}$ is given by:*

$$C_l \boldsymbol{\delta} \leq \max_{\substack{D,g,h,\\ f \triangleq \{e,D,g,h\},\\ f\in\mathcal{F}, X\in\mathcal{X}}} \; \min_{\substack{\mathsf{D},\mathsf{g},\mathsf{h},\\ \mathsf{f}\triangleq\{e,\mathsf{D},\mathsf{g},\mathsf{h}\},\\ \mathsf{f}\in\mathcal{F}}} |f(X) - \mathsf{f}(X)| \leq C_u \frac{\boldsymbol{\delta}}{\boldsymbol{\beta}}, \quad (4)$$

where $f(X) = h(g^{\otimes D(X)}(e(x_1), \ldots, e(x_L)))$, and $\mathsf{f}(X) = \mathsf{h}(\mathsf{g}^{\otimes \mathsf{D}}(e(x_1), \ldots, e(x_L)))$, with general smoothness and structural assumptions, and universal constants $C_l, C_u > 0$.

This result provides an upper and lower bound on the approximation error. Even if we select the best possible fixed **cDAG** D, and corresponding span processor g and read-out function h (the $\min$ over $\{\mathsf{D}, \mathsf{g}, \mathsf{h}\}$ for the approximation $\mathsf{f} \in \mathcal{F}$), there are compositional functions f such that, for some input X (the $\max$ over $\{D, g, h\}$ for $f \in \mathcal{F}$, and the input $X \in \mathcal{X}$), the approximation error is at least $\mathcal{O}(\boldsymbol{\delta})$. The approximation is bounded from above by $\mathcal{O}(\boldsymbol{\delta}/\boldsymbol{\beta})$; note that the maximum relative LoI $\boldsymbol{\beta} \in [1/L, 1]$. This indicates that function classes with large $\boldsymbol{\delta}$ are hard to approximate with an input-agnostic **cDAG** of the same complexity. Furthermore, for the same $\boldsymbol{\delta}$, smaller $\boldsymbol{\beta}$ worsen the upper bound. Overall, function classes with small $\boldsymbol{\delta}$, or moderate $\boldsymbol{\delta}$ with large $\boldsymbol{\beta}$ can be approximated well with input-agnostic **cDAG**s. As discussed earlier, for function classes of particular interest to us, with moderately high $\boldsymbol{\delta}$ and high $\boldsymbol{\beta}$, input-agnostic **cDAG**s do not provide promising approximation, though larger $\boldsymbol{\beta}$ is more favorable. This result makes precise the intuition that input-agnostic **cDAG**s are not expressive enough to appropriately approximate functions with input-dependent **cDAG**s.

We also study systematic generalization within our framework. We consider a special class of compositional functions with a maximum out-degree $q = 1$ in the **cDAG**, inducing "cleanly-separable" sub-structures, and further focus on the case with maximum in-degree $k = 2$ and a single sink ($m = 1$) in the **cDAG**. Given a class of (compositional) functions $\mathcal{F}$, and training data S, generalization guarantees bound the difference between the true risk $R(\hat{f}) = \mathbb{E}_{(X,y)}\ell(y, \hat{f}(X))$ of a learned model $\hat{f} \in \mathcal{F}$ and its empirical risk $R_N(\hat{f})$. Here we present a condensed result; see Appendix D in the supplement for details.

Theorem 2 (condensed)**.** *Consider a $(2, 1, 1, \boldsymbol{\delta}, \boldsymbol{\beta})$-compositional function class $\mathcal{F}$, input $X \in \mathcal{X}$ of length L, a training set S of N samples from a ground-truth function $f \in \mathcal{F}$ with components e, D, g, h. Assume that the token encoder e and the **cDAG** function D are given, and we only learn the span encoder $\hat{g}$ and the read-out function $\hat{h}$ to get $\hat{f} \triangleq (e, D, \hat{g}, \hat{h})$. Then, with probability at least $1 - \xi$,*

$$\left| R(\hat{f}) - R_N(\hat{f}) \right| \leq \gamma \boldsymbol{\delta} C_N \left(1 + 2N \sqrt{\frac{2\log(2/\xi)}{N}} \right), \quad (5)$$

where C_N is a quantity dependent on N and $\xi \in (0, 1)$.

If the model class $\mathcal{F}$ is expressive, the empirical risk $R_N(\hat{f})$ can be small. If $C_N \sim O(1/N)$, then we recover the standard $O(1/\sqrt{N})$ rate. However, if the compositional complexity, $\boldsymbol{\delta}$, is high, this bound does not provide a favorable systematic generalization guarantee, highlighting that, even with cleanly-separable compositions ($q = 1$), our proposed compositional complexity is directly tied to the systematic generalization.

5 Conclusion

In this paper, we proposed a precise novel definition of compositional functions, separating out the neural and symbolic parts, and a consequent compositional complexity, which explicitly quantifies the complexity in the (symbolic) computation trace or the **cDAG** of any compositional function. We demonstrated how existing models, such as recurrent, convolutional or attention-based ones, fit into our proposed definition, allowing us to compute and compare their compositional complexities. We categorized models into those with input-dependent compositions and those with input-agnostic ones, and established theoretical guarantees on the (in)ability of input-agnostic compositions to approximate input-dependent ones. Based on this theoretical framework, we also establish compositional generalization guarantees for learned compositional functions, rigorously connecting our proposed notion of compositional complexity to systematic generalization.

[2] We assume that the token encoder e is same for both f, f.

References

[Andreas, 2018] Jacob Andreas. Measuring compositionality in representation learning. In *International Conference on Learning Representations*, 2018.

[Bowman *et al.*, 2016] Samuel Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, 2016.

[Csordás *et al.*, 2021] Róbert Csordás, Kazuki Irie, and Juergen Schmidhuber. The devil is in the detail: Simple tricks improve systematic generalization of transformers. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 619–634, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[Drozdov *et al.*, 2023] Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. Compositional semantic parsing with large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[Dziri *et al.*, 2023] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36, 2023.

[Garcez *et al.*, 2022] Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Luis C Lamb, Leo de Penning, BV Illuminoo, Hoifung Poon, and COPPE Gerson Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation. *Neuro-Symbolic Artificial Intelligence: The State of the Art*, 342(1):327, 2022.

[Gordon *et al.*, 2019] Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. Permutation equivariant models for compositional generalization in language. In *International Conference on Learning Representations*, 2019.

[Gupta *et al.*, 2021] Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 39–52. Association for Computational Linguistics, 2021.

[Hupkes *et al.*, 2020] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: how do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

[Jarvis *et al.*, 2023] Devon Jarvis, Richard Klein, Benjamin Rosman, and Andrew M Saxe. On the specialization of neural modules. In *The Eleventh International Conference on Learning Representations*, 2023.

[Keysers *et al.*, 2019] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*, 2019.

[Kim and Linzen, 2020] Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, 2020.

[Klinger *et al.*, 2020] Tim Klinger, Dhaval Adjodah, Vincent Marois, Josh Joseph, Matthew Riemer, Alex 'Sandy' Pentland, and Murray Campbell. A study of compositional generalization in neural models. *arXiv preprint arXiv:2006.09437*, 2020.

[Lake and Baroni, 2018] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pages 2873–2882. PMLR, 2018.

[Li *et al.*, 2019] Yuanpeng Li, Liang Zhao, Jianyu Wang, and Joel Hestness. Compositional generalization for primitive substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4284–4293, 2019.

[Lin *et al.*, 2022] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

[Liu *et al.*, 2020] Qian Liu, Shengnan An, Jian-Guang Lou, Bei Chen, Zeqi Lin, Yan Gao, Bin Zhou, Nanning Zheng, and Dongmei Zhang. Compositional generalization by learning analytical expressions. *Advances in Neural Information Processing Systems*, 33, 2020.

[Liu *et al.*, 2021] Chenyao Liu, Shengnan An, Zeqi Lin, Qian Liu, Bei Chen, Jian-Guang Lou, Lijie Wen, Nanning Zheng, and Dongmei Zhang. Learning algebraic recombination for compositional generalization. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1129–1144, 2021.

[Nye *et al.*, 2020] Maxwell Nye, Armando Solar-Lezama, Josh Tenenbaum, and Brenden M Lake. Learning compositional rules via neural program synthesis. *Advances in Neural Information Processing Systems*, 33:10832–10842, 2020.

[Ontanon *et al.*, 2022] Santiago Ontanon, Joshua Ainslie, Zachary Fisher, and Vaclav Cvicek. Making transformers solve compositional tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3591–3607, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[Pagin and Westerståhl, 2010a] Peter Pagin and Dag Westerståhl. Compositionality I: Definitions and variants. *Philosophy Compass*, 5(3):250–264, 2010.

[Pagin and Westerståhl, 2010b] Peter Pagin and Dag Westerståhl. Compositionality II: Arguments and problems. *Philosophy Compass*, 5(3):265–282, 2010.

[Partee, 1995] Barbara Partee. Lexical semantics and compositionality. *An invitation to cognitive science. Part I: Language*, 1:311–360, 1995.

[Ram *et al.*, 2023] Parikshit Ram, Tim Klinger, and Alexander G. Gray. How compositional is a model? In *International Joint Conference on Artificial Intelligence 2023 Workshop on Knowledge-Based Compositional Generalization*, 2023.

[Ram *et al.*, 2024] Parikshit Ram, Tim Klinger, and Alexander G. Gray. What makes Models Compositional? A Theoretical View: With Supplement. *arXiv preprint arXiv:2405.02350*, 2024.

[Rosenbaum *et al.*, 2019] Clemens Rosenbaum, Ignacio Cases, Matthew Riemer, and Tim Klinger. Routing networks and the challenges of modular and compositional computation. *arXiv preprint arXiv:1904.12774*, 2019.

[Russin *et al.*, 2019] Jake Russin, Jason Jo, Randall C O'Reilly, and Yoshua Bengio. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*, 2019.

[Sarker *et al.*, 2021] Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-symbolic artificial intelligence. *AI Communications*, 34(3):197–209, 2021.

[Shen *et al.*, 2018] Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*, 2018.

[Shen *et al.*, 2019] Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*, 2019.

[Sikarwar *et al.*, 2022] Ankur Sikarwar, Arkil Patel, and Navin Goyal. When can transformers ground and compose: Insights from compositional generalization benchmarks. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 648–669, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[Socher *et al.*, 2010] Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 deep learning and unsupervised feature learning workshop*, volume 2010, pages 1–9, 2010.

[Tai *et al.*, 2015] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, 2015.

[Tay *et al.*, 2020] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. *Proceedings of ICML*, 2020.

[Tay *et al.*, 2022] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6), 2022.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wiedemer *et al.*, 2023] Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. Compositional generalization from first principles. *Advances in Neural Information Processing Systems*, 36, 2023.

[Zadrozny, 1994] Wlodek Zadrozny. From compositional to systematic semantics. *Linguistics and philosophy*, 17:329–342, 1994.

[Zhou *et al.*, 2023] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.