Exploring Explainable Compositionality of LLMs: A Program-Generation Perspective

Anonymous ACL submission

Abstract

Compositional generalization tests are often used to estimate the compositionality of LLMs. However, compositional generalization tests (1) do not focus on the explanations of LLMs for their fitted functions and (2) use consistency with a fixed function on a pre-partitioned test set as a criterion, hindering the acquisition of explainable and convincing estimation and analysis of the compositionality of LLMs. In this work, we propose a program-generation perspective that takes the programs generated by LLMs as externalized explanations and provides estimates of the compositionality of LLMs with the help of complexity-based theory. The perspective addresses the explainability limitations of compositional generalization tests and provides a new way to analyze the compositionality characterization of LLMs. We conduct experiments and analysis of existing advanced LLMs based on this perspective on a string-to-grid task, and find various compositionality characterizations and compositionality deficiencies exhibited by LLMs.

1 Introduction

Compositionality is a concept that originates in the philosophy of language. It is a property that a language has to a certain extent and can be expressed as "the meaning of a complex expression is determined by its structure and the meanings of its constituents" (Pelletier, 1994; Janssen and Partee, 1997; Szabó, 2004; Pagin and Westerståhl, 2010). In machine learning, the concept of compositionality is generalized to the mapping of inputs to outputs, suggesting that the output is determined by the meanings of the components of the input and the form in which the components are combined (Lake and Baroni, 2018; Hupkes et al., 2020). In the NLP domain, many tasks involve mappings with significant compositionality, such as semantic parsing (Keysers et al., 2020), data-to-text generation (Xu and Wang, 2024), compositional reasoning (Li et al., 2024), etc.

For a task that involves mappings with compositionality, if a model can recognize the compositionality of the mappings and utilize it, then the model will be able to correctly map the inputs made up of components to the outputs, as long as it knows the meaning of the components. This ability to recognize the compositionality of the mappings and utilize it is called the model's compositionality. Models' compositionality characterizes an effective form of reaching out-of-distribution generalization (Bahdanau et al., 2019) and this form is typical in human intelligence (Dehaene et al., 2022). Therefore, the compositionality of models is an important research topic from both practical and cognitive perspectives (Hupkes et al., 2022).

The research on models' compositionality has long been controversial, and the controversy focuses on how to properly measure a model's compositionality and whether the existing paradigms enable models to develop sufficient compositionality. In the NLP domain, a widely used approach to study the compositionality of language models on specific tasks is to conduct compositional generalization tests. The essence of the compositional generalization test is to partition the training and test sets with compositional differences, and then test the trained model's performance on the test set. After the emergence of large language models (LLMs), compositional generalization tests are still widely used under in-context learning for LLMs that are difficult to fine-tune directly.

The results of the compositional generalization test are intuitively suitable as a reflection of the compositionality of LLMs. However, compositional generalization tests have limitations regarding explainability, mainly in terms of (1) the lack of attention to the LLMs' explanation of their fitted functions, and (2) the lack of explainability in using consistency with a fixed function on a pre-partitioned test set as a criterion. The limitations make it difficult to obtain convincing estimates and analyses of the LLMs' compositionality, hindering more in-depth research on the explainable compositionality of the LLMs.

To solve this problem, we propose a program-generation perspective for the estimation and analysis of the compositionality of LLMs. In this perspective, we take the program generated by LLMs as an explanation of their fitted functions and draw on complexity-based theory to give an estimate of the compositionality of LLMs based on the explanation. By externalizing the explanation and appropriately quantifying the compositionality reflected in the explanation, this perspective addresses the explainability limitations of compositional generalization tests. The perspective is consistent with intuitions about compositionality and generalization, and provides new ways to characterize the compositionality of LLMs.

Based on this perspective, we experiment and analyze advanced LLMs including reasoning models and non-reasoning models on a simple string-to-grid task. We identify different compositionality characterizations exhibited by LLMs, and compositionality defects of LLMs in various situations.

2 Compositional Generalization Tests

In this section, we introduce the formulation of compositional functions and compositional generalization tests. We discuss the limitations of compositional generalization tests in terms of explainability.

2.1 Formulation

Following the formulation in Wiedemer et al. (2023), a compositional function f transforms K independent input components into K output components, and then combines these output components into an output. Formally, the K independent input components are K sets $C_1, ..., C_K$, where $C_k = \{v_{k,1}, ..., v_{k,U}\}$ denotes the U possible values of the k-th input component. For a value $c_k \in C_k$, the transformation function $\phi_k : C_k \to R_k$ transforms it into the output component r_k. The combination function $g : R_1 \times \cdots \times R_K \to Y$ combines the components into the output y. We define $X = C_1 \times \cdots \times C_K$ to denote the set containing all possible inputs. Given the input $x = (c_1, ..., c_K) \in X$, the compositional function $f : X \to Y$ can be expressed as:

$$f(x) = g(\phi_1(c_1), ..., \phi_K(c_K)) \qquad (1)$$

For an unknown compositional function f, compositional generalization requires that the model be able to map unseen combinations of component values to expected outputs after seeing all the component values and some combinations of component values mapped to the outputs. Compositional generalization tests typically follow the training-test paradigm. In this paradigm, we divide X into two disjoint subsets X_S and X_T that satisfy $\forall\, v_{k,j}\ ,\ \ \exists\, x \in X_S\ ,\ x_k = v_{k,j}$, and generate training set $S = \{(x, f(x)) \mid x \in X_S\}$ and test set $T = \{(x, f(x)) \mid x \in X_T\}$. The division is usually based on minimizing the degree to which combinations of components in T are visible in S (Keysers et al., 2020; Kim and Linzen, 2020). After a model is trained on the training set S, the model's accuracy on the test set T is used to measure the model's compositional generalization performance. For LLMs that are difficult to fine-tune directly, each test of $x \in X_T$ is usually performed independently by extracting a subset of S that covers the values in x to be input to the LLMs as a demonstration of in-context learning.

2.2 Limitations of Tests

It is intuitively appropriate to use the model's compositional generalization performance to reflect the model's compositionality. However, the compositional generalization test has the following limitations in terms of explainability:

(L1) The model's explanation of the function f^* it fits cannot be obtained simply from the mapping results, preventing a convincing analysis of the model's compositionality. In compositional generalization tests, we only observe the mapping results output by the model without focusing on the process of generating the mapping results. However, by simply observing the mapping results, we cannot obtain an explanation of the model for the function f^* it fits. In this case, we cannot provide a convincing analysis of the model's compositionality based on the model's explanation of its fitted function f^*. For example, we cannot convincingly capture what exactly the model recognizes as the samples' compositionality and analyze how it differs from our expectations, making it difficult to explain the model's errors in compositional generalization tests.

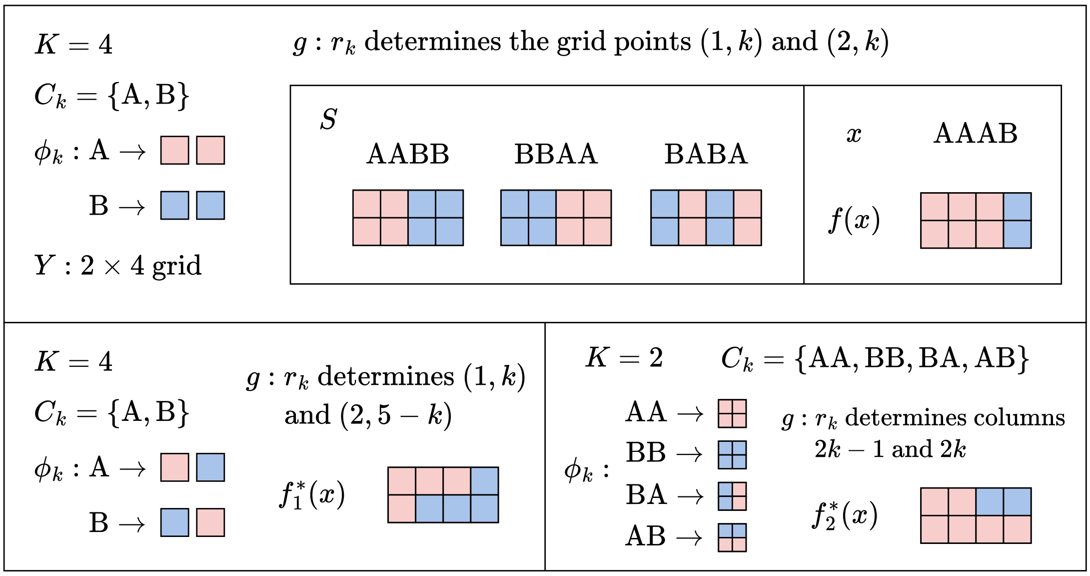

Figure 1: The above shows a compositional function f that maps a 4-bit A / B string to a 2×4 grid. Models 1 and 2 fit functions f_1^* and f_2^* which fit S but are inconsistent with f on x. **L1**: Based only on the mapping results output by the model, the explanation for f^* as in the figure cannot be obtained for analysis. **L2**: Even if the model fits a function inconsistent with f on x, it may have a sufficiently compositional (not clearly defined) explanation, e.g., the explanation of f_1^* in the figure is intuitively sufficiently compositional while the explanation of f_2^* is not.

(L2) Using consistency with a fixed function f on a pre-partitioned test set as a criterion lacks explanability and may lead to unconvincing estimates of the model's compositionality. For a training set S, the function that can fit it is not unique. Compositional generalization tests use f as a fixed criterion, requiring the model to perform consistently with f on the test set. The reason for choosing f as a fixed criterion is usually that the explanation of f is sufficiently compositional from human intuition, but we lack a clear definition of what is "compositional". In this case, the estimate of the model's compositionality lacks explainability: even if the model's performance on $x \in T$ is inconsistent with f, its fitted function f^* may, under some definition, be fairly "compositional" in its explanation, and the estimate is therefore unconvincing. To solve this problem, it is necessary to move away from the paradigm that partitions the training and test sets and evaluates the consistency with f on the test set. We need to clearly define the compositionality reflected in the explanation and give an estimate of the model's compositionality through the model's explanation of its fitted function. To do this, **L1** first needs to be addressed.

Figure 1 provides a specific example illustrating **L1** and **L2**. The development of the performance of LLMs has made it possible to direct LLMs to export their explanations, which motivates us to consider a more explainable perspective for measuring and analyzing the compositionality of LLMs to address both **L1** and **L2**.

3 Program-Generation Perspective

In this section, we propose the program-generation perspective for the estimation and analysis of the compositionality of LLMs. We introduce the rationale and formulation of the perspective, and the characterization of the compositionality of LLMs provided by quantitative metrics. We show that this perspective addresses the limitations of compositional generalization tests in terms of explainability.

3.1 Rationale

The key to addressing **L1** and **L2** is that (1) we need to be explicit about the explanation of the LLMs for the functions they fit, and (2) we need a method for properly estimating the compositionality of the LLMs based on their explanation.

Since it is difficult to analyze the explanations of the LLMs from the internal states, we use externalization, i.e., we ask the LLMs to directly output their explanations of the fitted functions. We want the explanation to be presented in a formal language with unambiguity, and the LLMs need no additional guidance for the generation of this formal language. Therefore, we choose a common programming language as the formal language of

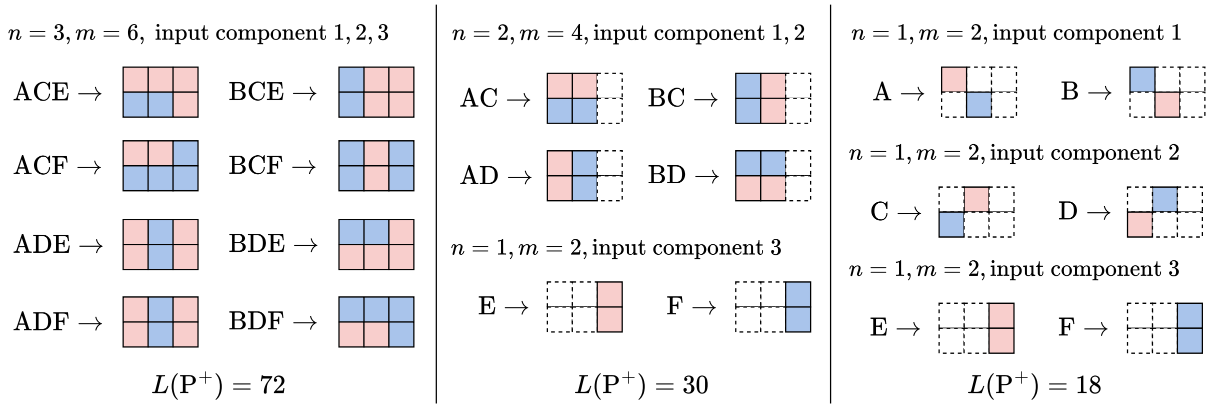

Figure 2: Examples of the mapping table of P^+ (3 atomic input components, 6 atomic output components, 8 samples). We group mappings involving the same atomic input components and mark the involved atomic output components with colors. The leftmost and rightmost examples demonstrate zero and sufficient compositionality.

the explanation and ask the LLMs to output the program as the explanation. Specifically, for the set $D = \{(x_i, y_i) \mid x_i \in X, y_i = f(x_i)\}_{i=1}^{d}$ generated by a compositional function f containing d samples that cover all possible input component values, we ask the LLMs to output a program P satisfying that for any $i \in \{1, ..., d\}$, the program P outputs y_i on input x_i.

To estimate the compositionality of LLMs via the program P, we introduce the complexity-based theory of compositionality proposed by Elmoznino et al. (2025). The theory is based on Kolmogorov Complexity $\mathcal{K}$ (Kolmogorov, 1965) for a quantitative definition of the compositionality of mappings from a compression perspective. For object lists I and O, $\mathcal{K}(O)$ denotes the length of the shortest program (in a certain programming language) that outputs O, and $\mathcal{K}(O|I)$ denotes the length of the shortest program that outputs O with input I. Let $D_X = \{x_i\}_{i=1}^{d}$ and $D_Y = \{y_i\}_{i=1}^{d}$ be lists of x_i and y_i in D, respectively. In this theory, the compositionality of the set D (regarded as a mapping from D_X to D_Y) is defined as $\frac{\mathcal{K}(D_Y)}{\mathcal{K}(D_Y|D_X)}$, which intuitively means the extent to which the representation of D_Y can be compressed using D_X.

Although $\mathcal{K}$ is not computable, its upper bound can be estimated. The compositionality of an LLM can be characterized as how small an estimate of the upper bound on $\mathcal{K}(D_Y|D_X)$ is provided by the program P that the LLM generates, as smaller estimates indicate a stronger degree of compression. The most direct upper bound estimate provided by a correct P is the length itself. However, the length of P is affected by many non-essential factors (e.g., formatting, naming, different description of the same process, etc.), and P may be incorrect on D (i.e., for some input x_i, the output is not y_i), so the upper bounds provided by different P with their lengths may lack comparability. We can transform P into a hypothetical program P^+ in a uniform programming paradigm such that P^+ is correct on D and the upper bound estimates provided by different P^+ are comparable. The upper bound estimates provided by P^+ can then be used as a basis for estimating the compositionality of LLMs.

3.2 Formulation

Suppose a sample contains N atomic input components and M atomic output components. A hypothetical program P^+ contains a mapping table consisting of z mappings. The z-th mapping maps the values of n_z input components to the values of m_z atomic output components. Using the mapping table, P^+ transforms the input into output components and combines them into an output by a fixed algorithm. Assuming that the values of all atomic input and output components are programmed with length 1, we have that the length of P^+ is $w_1 \cdot \sum_{z=1}^{Z}(n_z + m_z) + w_2$, where w_1, w_2 are constants that are consistent for any P^+. Thus we define the size of the mapping table as a comparable metric for the estimates provided by P^+:

$$L(\mathrm{P}^+) = \sum_{z=1}^{Z}(n_z + m_z) \qquad (2)$$

Figure 2 illustrates the meaning of $L(\mathrm{P}^+)$. By parsing P for locations involving values of the input and output components (see Appendix A for details), we can get the metric $L(\mathrm{P}^+)$ of its corresponding P^+. We also need to check the correctness of P on D. If there are E errors (i.e.,

$E = \sum_{i=1}^{d}[\mathrm{P}(x_i) \neq y_i]$), then E mappings directly corresponding to the error samples (all N atomic input components mapped to all M atomic output components) will need to be added to the mapping table to make P^+ correct, and so $L(\mathrm{P}^+)$ increases by $(N+M)E$. We thus obtain a metric for the estimates provided by P:

$$L(\mathrm{P}) = L(\mathrm{P}^+) + (N+M)E \quad (3)$$

For a P^+ that is correct on D, there are two bounds on the size of its mapping table: (1) each atomic input component corresponds to a fixed set of atomic output components (mutually disjoint and the union is all atomic output components), each mapping is a mapping of the value of an atomic input component to the value of its corresponding atomic output components, and the mapping table size $L_{\mathrm{s}} = U(N+M)$ corresponds to sufficient compositionality; (2) the mapping table contains d mappings directly corresponding to d samples, and the mapping table size $L_{\mathrm{z}} = d(N+M)$ corresponds to zero compositionality. Figure 2 illustrates examples of the two bounds. If $d > U$ (i.e., at least one of the values appears more than once in D), we can normalize the metric:

$$\mathcal{C}(\mathrm{P}) = 100 \cdot \frac{L_{\mathrm{z}} - \mathrm{Clip}(L(\mathrm{P}), L_{\mathrm{z}}, L_{\mathrm{s}})}{L_{\mathrm{z}} - L_{\mathrm{s}}} \quad (4)$$

where Clip takes the value $L(\mathrm{P})$ when $L(\mathrm{P}) \in [L_{\mathrm{z}}, L_{\mathrm{s}}]$, and otherwise is the one closer to $L(\mathrm{P})$ among L_{z} and L_{s}. We have $\mathcal{C}(\mathrm{P}) \in [0, 100]$. A larger metric $\mathcal{C}(\mathrm{P})$ represents a smaller estimate of $\mathcal{K}(D_Y|D_X)$ provided by P, reflecting a stronger compositionality of the LLM.

3.3 Characterization

In the program-generation perspective, $L(\mathrm{P}^+)$ is consistent with our intuition about compositionality, as smaller $\sum n_z$ indicates that the LLM is more aware of the independence of the input components, and smaller $\sum m_z$ indicates that the LLM more accurately identifies the output components that are influenced by the input components. Also, the characterization of the degree of compression by $L(\mathrm{P}^+)$ is consistent with our intuition about generalization. $L(\mathrm{P})$ further takes the number of errors E into account and is normalized to the metric $\mathcal{C}(\mathrm{P})$. The combination of $L(\mathrm{P}^+)$ and E can provide a holistic and relative characterization of the compositionality of LLMs into three types:

(T1) Low $L(\mathrm{P}^+)$ and Low E. LLMs exhibit sufficient compositionality: they adequately and correctly capture the compositionality of the samples and utilize it to describe D.

(T2) High $L(\mathrm{P}^+)$ and Low E. LLMs do not adequately capture the compositionality of the samples and therefore choose to low-compressively but high-correctly describe D (e.g., simply using all samples directly in the program).

(T3) Low $L(\mathrm{P}^+)$ and High E. LLMs do not adequately capture the compositionality of the sample, but still try to highly compress the description of D, leading to a highly erroneous description.

Under this characterization, $L(\mathrm{P}^+)$ and E are two dimensions that characterize the degree of compression and compression loss, respectively. The high $\mathcal{C}(\mathrm{P})$ exhibited by **T1** can be thought of as a low-loss high compression of samples through their compositionality. The low $\mathcal{C}(\mathrm{P})$ exhibited by **T2** and **T3** both manifestations of the inability to correctly capture the compositionality of the samples, but they exhibit different biases: **T2** is biased towards low loss and **T3** is biased towards high compression.

3.4 Addressing Limitations

The program-generation perspective addresses **L1** and **L2** of compositional generalization tests:

(1) In this perspective, we use the program output by the LLMs as an unambiguous explanation of the function they fit on D. Based on the explanation, we are able to provide a more convincing analysis of the compositionality of LLMs, thus addressing **L1**.

(2) In this perspective, we quantify the compositionality of LLMs reflected in the explanation program P as metrics, which draw on complexity-based theory to give a clear definition of how compositional an explanation is. Although we have a function f to generate D, we do not need to perform a training-test partition, and the quantitative metrics do not involve any consistency measure with the explanation of f. This perspective moves away from the paradigm of using consistency with a fixed function on the test set as a criterion, and provides a more convincing estimate of the compositionality of LLMs based on explanations, thus addressing **L2**.

4 Experiments and Analysis

4.1 Experimental settings

Task Formulation. The input of the compositional function is a string of length $N = 4$ and the output

	Horizontal			Block			Vertical			Random		
	$L(\mathrm{P}^{+})$	E	$\mathcal{C}(\mathrm{P})$	$L(\mathrm{P}^{+})$	E	$\mathcal{C}(\mathrm{P})$	$L(\mathrm{P}^{+})$	E	$\mathcal{C}(\mathrm{P})$	$L(\mathrm{P}^{+})$	E	$\mathcal{C}(\mathrm{P})$
DeepSeek-R1	43.23	1.33	89.80	254.43	3.63	7.62	254.77	6.00	0.00	270.00	5.13	0.00
QwQ-Plus	71.33	7.30	44.00	118.33	11.23	2.86	**95.97**	9.17	23.31	**181.27**	9.20	0.00
o1-mini	49.43	0.30	94.49	215.23	3.03	19.38	280.87	2.23	4.29	278.53	3.20	0.00
o3-mini	46.73	**0.27**	**95.69**	150.87	0.47	**57.07**	243.53	0.07	27.31	318.13	**0.07**	0.67
Gemini-2.5	45.83	0.70	92.92	197.03	**0.13**	42.96	234.90	**0.00**	**30.39**	292.00	0.10	**10.00**
Claude-3.7	**40.27**	2.20	84.67	**89.17**	6.40	47.57	216.30	5.10	14.71	192.33	8.63	3.52
DeepSeek-V3	165.27	**3.20**	42.87	241.53	6.60	0.00	258.87	5.13	0.00	239.70	5.90	3.33
Qwen-Max	**44.57**	8.53	**46.67**	**50.70**	15.03	**0.48**	**71.80**	14.93	0.00	**62.00**	14.77	0.00
GPT-4o	46.67	15.70	0.00	51.80	15.90	0.00	90.63	15.90	0.00	111.33	15.93	0.00
Gemini-2.0	189.60	7.17	7.77	255.93	**4.50**	0.43	286.13	**2.13**	**0.67**	295.00	**1.17**	**3.71**
Claude-3.5	118.43	14.10	6.69	135.03	15.10	0.00	120.00	16.00	0.00	133.97	15.43	0.00

Table 1: Results of the base experiment.

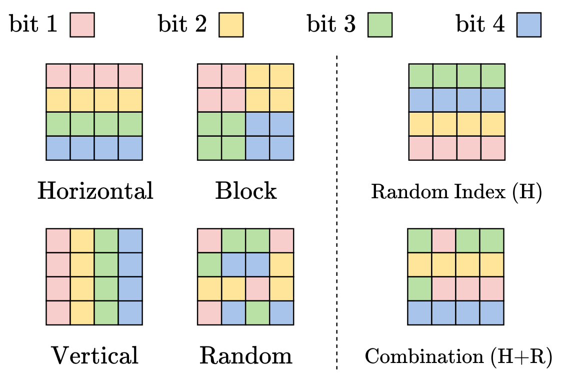

Figure 3: An illustration of the experimental settings. The color of each grid point indicates the input string bit that determines its value.

is a 4×4 grid ($M = 16$). Each grid point has 2 possible values $[\cdot, *]$. Each bit of the string has $U = 2$ possible values, and the possible values of the i-th bit are the $(2i - 1)$-th and $(2i)$-th uppercase letters. Each bit of the string determines the value of 4 grid points, and the set of grid points determined by each bit of the string is mutually exclusive. The value of any grid point differs when the value of the input bit that determines it differs. All possible $d = 16$ samples generated by the compositional function are provided to the LLMs as D and the LLMs are asked to generate a Python program to describe D.

Data Generation. Our base experiment consists of four different compositional function settings: (1) Horizontal (the i-th bit determines the i-th row), (2) Block (the i-th bit determines the i-th 2×2 subgrid), (3) Vertical (the i-th bit determines the i-th column), and (4) Random (each bit determines 4 random grid points). For each setting, we sample 30 different compositional functions for data generation and report the average results of LLMs on the task. Our extended experimental settings, including random index and setting combination, are discussed further in 4.3. Figure 3 shows an illustration of the settings.

Evaluated LLMs. The LLMs we evaluate include the reasoning models: DeepSeek-R1-0120 (DeepSeek-AI et al., 2025a), QwQ-Plus-0305 (Qwen-Team, 2025), o1-mini-2024-09-12 (OpenAI et al., 2024b), o3-mini-2025-01-31 (OpenAI, 2025), Gemini-2.5-pro-exp-03-25 (Google, 2025), Claude-3.7-Sonnet-20250219 (Anthropic, 2025), and the non-reasoning models: DeepSeek-V3-0324 (DeepSeek-AI et al., 2025b), Qwen-Max-0125 (Yang et al., 2024), GPT-4o-2024-08-06 (OpenAI et al., 2024a), Gemini-2.0-flash (Google, 2024), and Claude-3.5-Haiku-20241022 (Anthropic, 2024).

4.2 Base Experiment

Table 1 shows the results of the base experiment.

4.2.1 Compositionality Characterization

The non-reasoning models we evaluate exhibit fairly low compositionality in most cases, and only DeepSeek-V3 and Qwen-Max exhibit relatively high compositionality in the Horizontal setting. Relatively among the non-reasoning models: (1) Qwen-Max, GPT-4o, and Claude-3.5 are characterized as **T3**. Although they exhibit strong compression, their extremely high error rate means that their descriptions of D are almost completely incorrect. (2) Deepseek-V3 and Gemini-2.0 are characterized as **T2**. They have a relatively high degree of correctness in describing D, but also a relatively low degree of compression.

The reasoning models we evaluate generally exhibit stronger compositionality than non-reasoning models in settings other than Random. The stronger

	Random Index (H)			Setting Combination (H+R)		
	$L(\mathrm{P}^+)$	E	$\mathcal{C}(\mathrm{P})$	$L(\mathrm{P}^+)$	E	$\mathcal{C}(\mathrm{P})$
DeepSeek-R1	174.07 (+130.83)	4.17 (+2.83)	26.54 (-63.26)	187.30 (+30.68)	3.57 (+0.33)	27.43 (-17.47)
QwQ-Plus	138.23 (+66.90)	10.97 (+3.67)	0.00 (-44.00)	147.60 (+21.30)	9.83 (+1.58)	4.00 (-18.00)
o1-mini	138.27 (+88.83)	1.27 (+0.97)	56.69 (-37.80)	176.60 (+12.62)	2.67 (+0.92)	34.26 (-12.98)
o3-mini	106.90 (+60.17)	**0.50** (+0.23)	73.20 (-22.49)	84.73 (-97.70)	0.77 (+0.60)	**78.79** (+30.61)
Gemini-2.5	87.57 (+41.73)	0.93 (+0.23)	76.58 (-16.33)	205.53 (+36.62)	**0.73** (+0.33)	38.98 (-12.48)
Claude-3.7	**40.23** (-0.03)	3.13 (+0.93)	**79.04** (-5.63)	**62.57** (-53.73)	3.80 (-1.62)	66.39 (+22.30)
DeepSeek-V3	145.40 (-19.87)	8.57 (+5.37)	**12.85** (-30.02)	207.43 (+4.95)	5.27 (+0.72)	**9.89** (-13.21)
Qwen-Max	51.00 (+6.43)	14.83 (+6.30)	1.38 (-45.29)	58.93 (+5.65)	13.67 (+2.02)	6.67 (-16.67)
GPT-4o	**45.33** (-1.33)	15.80 (+0.10)	0.00 (-0.00)	**58.20** (-20.80)	15.80 (-0.02)	0.00 (-0.00)
Gemini-2.0	267.10 (+77.50)	**3.83** (-3.33)	1.81 (-5.96)	267.53 (+25.23)	**4.07** (-0.10)	0.43 (-5.32)
Claude-3.5	116.37 (-2.07)	15.73 (+1.63)	0.00 (-6.69)	113.60 (-12.60)	15.83 (+1.07)	0.00 (-3.35)

Table 2: Results of extended experiments. The amount of change compared to the results of the base experiment is shown in parentheses (left: compared to Horizontal; right: compared to the average of Horizontal and Random).

compositionality stems from maintaining a certain degree of compression at a generally lower number of errors. However, the reasoning models do not exhibit sufficient compositionality in settings other than Horizontal. In settings other than Horizontal, the reasoning models exhibit the following characterization in relative terms: (1) QwQ-Plus and Claude-3.7 are characterized as **T3**. They exhibit a high error rate and a high degree of compression. (2) Gemini-2.5 and o3-mini are characterized as **T2**. They exhibit a low error rate and a low degree of compression. (3) DeepSeek-R1 and o1-mini are characterized roughly between **T2** and **T3**.

4.2.2 Impact of the Settings

Since all possible samples are provided in D, for any bit of the input string, LLMs can theoretically find that the bit independently determines some grid points from multiple pairs of samples differing only on that bit, which is independent of the compositional function setting. However, the $\mathcal{C}(\mathrm{P})$ of LLMs differ clearly across settings and mostly follow the relation: Horizontal $>$ Block $>$ Vertical $>$ Random (except for QwQ-Plus which shows Vertical $>$ Block).

We hypothesize that the compositionality exhibited by LLMs on this task is influenced by how intuitive the sample's compositionality is. Of the four compositional function settings, the first three have a certain regularity and a similar intuition in the two-dimensional view, since the grid points determined by each bit of the string in these settings are connected in the two-dimensional plane. However, in the linear form of text input, for the continuity of the grid point positions determined by each bit of the string in the settings, we have Horizontal $>$ Block $>$ Vertical. As continuity declines, the intuition of the compositionality of samples may decline for LLMs, which are used to intuitively capturing by row. The Random setting, on the other hand, provides no intuition for LLMs at all.

4.3 Extended Experiments

Table 2 shows the results of the extended experiments.

4.3.1 Random Index

We conduct extended experiments with the Random Index setting on the Horizontal setting, where LLMs exhibit the strongest compositionality in the base experiment. In the base experiment, the i-th bit of the input string corresponds to the i-th row of the grid in the Horizontal setting. With the extended Random Index setting, the row corresponding to each bit is randomized.

The results show that the Random Index setting causes a severe weakening of the compositionality exhibited by the LLMs. For reasoning models, all models except Claude-3.7 exhibit high $L(\mathrm{P}^+)$ increases, indicating a reduction in compression. For reasoning models, all models except Claude-3.7 exhibit a high increase in $L(\mathrm{P}^+)$, indicating a decrease in compression; all models exhibit an increase in E, indicating an elevated compression loss, especially DeepSeek-R1 and QwQ-Plus. Among the reasoning models, Claude-3.7 exhibits the least decrease in compositionality and shows the strongest compositionality with the Random Index setting. Most of the non-reasoning models show a decrease in compositionality, approaching zero compositionality.

Although Random Index does not change the Horizontal pattern followed by each component mapping, the LLMs generally show a decline in compositionality, which is partly indicative of the LLMs' reliance on sequential correspondences for sample compositionality capture for this task, reflecting a deficiency in compositionality.

4.3.2 Setting Combination

We combine Horizontal and Random, the two settings where LLMs exhibit the strongest and weakest compositionality in the base experiment. Under the setting combination, two random rows in the grid use the Horizontal setting, and the other grid points use the Random setting.

For the reasoning models, compared to the average of the metrics in the two settings: (1) DeepSeek-R1, QwQ-Plus, o1-mini, and Gemini-2.5 exhibit elevated $L(\mathrm{P}^+)$ and E and decreased $\mathcal{C}(\mathrm{P})$. This means that when the compositionality of a portion of the sample's components (Random) is difficult to capture, their degree of compression and compression loss for all components are affected, even though the compositionality of the remaining components (Horizontal) is relatively easy for them to capture. This reflects a compositionality flaw of LLMs in that they have difficulty in independent compositionality capture for different sets of components. (2) Claude-3.7 and o3-mini exhibit elevated $\mathcal{C}(\mathrm{P})$, which suggests that they are somewhat capable of independent compositionality capture for the Horizontal component. In this case, even if they still exhibit low compression in the Random portion, there is a clear $L(\mathrm{P}^+)$ reduction brought about by the reduction of the component space. In addition, Claude-3.7 also exhibits a decrease in E, which indicates that its compression loss can also be reduced as the component space is reduced. The non-reasoning models mostly exhibit a decrease in $\mathcal{C}(\mathrm{P})$, approaching zero compositionality. Overall, many of the LLMs exhibit deficiencies in independent compositionality capture for different sets of components.

4.4 Qualitative Analysis

Figure 4 shows fragments of some of the programs generated by LLMs.

(1) There are some examples of high $\mathcal{C}(\mathrm{P})$ in settings other than Random. The output strings determined by a bit of the input string typically each correspond to at most one segment of a contiguous region within one row of the grid in linear form. This partly supports our hypothesis in 4.2.2.

(2) Typical examples of high $L(\mathrm{P}^+)$ and low E are simply enumerating all samples in all D.

T1: high $\mathcal{C}(\mathrm{P})$

Horizontal

```
"A": "**..", "B": "..**"
```

Block

```
if top_left == 'A':
    top_left_row1 = ".."
    top_left_row2 = "**"
else:  # assume 'B'
    top_left_row1 = "**"
    top_left_row2 = ".."
```

Vertical

```
# First row
if 'A' in input_letters:
    grid[0][0] = '*'
if 'C' in input_letters:
    grid[0][1] = '*'
if 'E' in input_letters:
    grid[0][2] = '*'
if 'G' in input_letters:
    grid[0][3] = '*'
```

Random Index

```
if "G" in letters:
    output[0] = ".*.*"
else:
    output[0] = "*.*."

if "C" in letters:
    output[1] = "..*."
else:
    output[1] = "**.*"
```

T2: high $L(\mathrm{P}^+)$, low E

```
"A D E G":
    "**.*\n*...\n****\n....",
"B D F H":
    "...*\n.***\n.*.*\n****",
"A C E H":
    ".**.\n*.*.\n*.*.\n*..*",
"B C F H":
    "..*.\n.***\n....\n****",
```

T3: low $L(\mathrm{P}^+)$, high E

```
for row in range(4):
    col = columns[row]
    line = []
    for c in range(4):
        if c == col:
            line.append('.')
        else:
            line.append('*')
```

Setting Combination

```
if l1 == 'A' and l3 == 'E':
    grid.append(".*..")
elif l1 == 'B' and l3 == 'E':
    grid.append("....")
...
if l2 == 'C':
    grid.append(".**.")
elif l2 == 'D':
    grid.append("*..*")
```

Figure 4: Examples of fragments of programs generated by LLMs.

Typical examples of low $L(\mathrm{P}^+)$ and high E are compression using simple algorithms not fully supported by D.

(3) High $\mathcal{C}(\mathrm{P})$ with Random Index setting is exemplified by the perception of sequential non-correspondence, and high $\mathcal{C}(\mathrm{P})$ with Setting Combination is exemplified by the perception of regions independently affected by different settings. In the extended experiments, typical examples of high E are still generating programs according to the Horizontal setting in the base experiment; typical examples of high $L(\mathrm{P}^+)$ are the same as in (2), arising from the inability to capture compositionality due to the extended settings.

5 Conclusion

In this work, we propose the program-generation perspective for estimating and analyzing the compositionality of LLMs. This perspective addresses the explainability limitations of compositional generalization tests and provides a new way to analyze the compositionality characterization of LLMs. Through experiments and analysis based on this perspective, we identify different compositionality characterizations and compositionality defects exhibited by existing advanced LLMs. This perspective provides support for the study of explainable compositionality of LLMs.

Limitations

The program-generation perspective is an exploratory perspective that still has limitations in its implementation:

(1) The perspective uses programs as an externalization of the interpretations of LLMs to allow for unambiguous parsing of the explanations. This requires that the LLMs under study have a certain level of program generation capability: on the task under consideration, the LLMs should at least be able to generate the correct program when provided with a complete explanation of the algorithm that generates D.

(2) To exclude the influence of non-essential factors, the perspective requires a parser to implement the conversion from the program to the estimate of the upper bound on $\mathcal{K}$ provided by it. Due to the diversity of the generated programs, it is difficult for the parser to cover all possible cases, potentially leading to bias in the conversion.

In this work, to minimize the impact of the above limitations on the results, we (1) pre-check the basic program generation capabilities of LLMs and (2) discover cases that the parser fails to cover and adjust the implementation through example testing and manual checking. We will continue to investigate how this perspective can be better applied to a wider range of models and tasks, and hope that the perspective can provide insights into explainable compositionality studies.

Ethics Statement

We comply with the license to use language models for scientific research purposes only. The datasets we construct will also be open source for scientific research purposes. The datasets we use do not contain any information that names or uniquely identifies individual people or offensive content.

The AI assistant we use in our work is Copilot (for simple code completion).

References

Anthropic. 2024. Claude 3.5 sonnet model card addendum.

Anthropic. 2025. Claude 3.7 sonnet system card.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron C. Courville. 2019. Systematic generalization: What is required and can it be learned? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng

Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025b. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

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

Eric Elmoznino, Thomas Jiralerspong, Yoshua Bengio, and Guillaume Lajoie. 2025. A complexity-based theory of compositionality. *Preprint*, arXiv:2410.14817.

Google. 2024. Introducing gemini 2.0: our new ai model for the agentic era.

Google. 2025. Gemini 2.5: Our most intelligent ai model.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *J. Artif. Intell. Res.*, 67:757–795.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2022. State-of-the-art generalisation research in NLP: a taxonomy and review. *CoRR*, abs/2210.03050.

Theo M.V. Janssen and Barbara H. Partee. 1997. Chapter 7 - compositionality. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 417–473. North-Holland, Amsterdam.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9087–9105. Association for Computational Linguistics.

Andrei N Kolmogorov. 1965. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1(1):1–7.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR.

Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. 2024. Understanding and patching compositional reasoning in llms. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 9668–9688. Association for Computational Linguistics.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024a. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024b. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI. 2025. Openai o3-mini system card.

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

Francis Jeffry Pelletier. 1994. The principle of semantic compositionality. *Topoi*, 13(1):11–24.

Qwen-Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Zoltán Gendler Szabó. 2004. Compositionality. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University.

Thaddäus Wiedemer, Prasanna Mayilvahanan, Matthias Bethge, and Wieland Brendel. 2023. Compositional generalization from first principles. In *Advances in Neural Information Processing Systems*, volume 36, pages 6941–6960. Curran Associates, Inc.

Ziyao Xu and Houfeng Wang. 2024. SPOR: A comprehensive and practical evaluation method for compositional generalization in data-to-text generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 604–621. Association for Computational Linguistics.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

A Details on obtaining $L(\mathrm{P}^+)$

To obtain the corresponding metric $L(\mathrm{P}^+)$ from the program P, we need to determine the $\sum n_z$ and $\sum m_z$ mapping table by the location of the values involving the input and output components in P. The contents of the comments are ignored.

A.1 Determination of $\sum n_z$

To determine $\sum n_z$, we need to determine which combinations of values of the input components used to determine the output are contained in P. A combination consists of values of input components that are (1) in a mapping of an explicit mapping table (dictionary), (2) on the same row, and (3) on a path in a nested structured tree of conditional judgments. For cases (2) and (3), it is considered to be used to determine the output when the row or the execution statements of the conditional judgments involve the values of the output components.

With the help of Python's AST tool, we are able to get all combinations of values of input components used to determine the output. The value of an input component may be on the right of an assignment statement and then affect a wider range through the variables on the left of the assignment statement. To handle this situation, we maintain the set of values of the input components involved for each variable due to assignment and utilize them when determining the values of the input components involved in the statement. For the nested structure of conditional judgments, we construct trees and obtain all possible paths and corresponding combinations by traversing them. For an *else* statement, we match it to the corresponding *if* and *elif* statements and treat it as containing one hypothetical value for each input bit involved in the *if* and *elif* statements.

The same combination may occur several times in P and can be generalized to the same mapping. Therefore, we count the total length of the values of the input components involved in mutually exclusive combinations as $\sum n_z$.

A.2 Determination of $\sum m_z$

$\sum m_z$ can theoretically be determined by finding the values of all output components in P and computing the length sum. However, we find that P sometimes expresses the determination of the output indirectly in other forms (e.g., storing the coordinates of the determined grid points), which occurs mainly in the explicit mapping table (dictionary). Therefore, we perform additional processing to count each atomic unit on the right side of the explicit mapping table as a value of an output component in $\sum m_z$.

A.3 Examples

Figure 5 shows two example code fragments. They both have $\sum n_z = 12$ and $\sum m_z = 48$.

Code fragment 1 mainly shows the case with conditional judgments. The first three code blocks contribute $2, 8, 2$ to $\sum n_z$, and $8, 16, 8$ to $\sum m_z$. All combinations of input component values in the last code block have already appeared in the second code block, so they are no longer counted in $\sum n_z$. The last code block contributes 16 to $\sum m_z$.

Code fragment 2 mainly shows the case with dictionaries. The four dictionaries contribute $16, 8, 16, 8$ to $\sum m_z$. The three dictionaries, except dictionary 3, contribute $8, 2, 2$ to $\sum n_z$.

```python
# First row is determined by the first letter (A or B)
if letters[0] == "A":
    row1 = ".*.*"
else:  # B
    row1 = "*.*."

# Second row is determined by the combination of second and fourth letters
if letters[1] == "C" and letters[3] == "G":
    row2 = "*.*."
elif letters[1] == "C" and letters[3] == "H":
    row2 = "...."
elif letters[1] == "D" and letters[3] == "G":
    row2 = "****"
else:  # D and H
    row2 = ".*.*"

# Third row is determined by the third letter (E or F)
if letters[2] == "E":
    row3 = ".***"
else:  # F
    row3 = "*..."

# Fourth row is determined by the combination of second and fourth letters
if letters[1] == "C" and letters[3] == "G":
    row4 = ".*.."
elif letters[1] == "C" and letters[3] == "H":
    row4 = "..*."
elif letters[1] == "D" and letters[3] == "G":
    row4 = "**.*"
else:  # D and H
    row4 = "*.**"
```

```python
# Mapping for first row (determined by first and third letters)
row1_patterns = {
    ("A", "E"): "...*",
    ("A", "F"): "*.*.",
    ("B", "E"): ".*.*",
    ("B", "F"): "***."
}

# Mapping for second row (determined by second letter)
row2_patterns = {
    "C": "****",
    "D": "...."
}

# Mapping for third row (determined by first and third letters)
row3_patterns = {
    ("A", "E"): "*.*.",
    ("A", "F"): "***.",
    ("B", "E"): "...*",
    ("B", "F"): ".*.*"
}

# Mapping for fourth row (determined by fourth letter)
row4_patterns = {
    "G": ".*.*",
    "H": "*.*."
}
```

Figure 5: Two examples of fragments of programs generated by LLMs. They both have $\sum n_z = 12$ and $\sum m_z = 48$.