Operads for compositional reasoning in LLMs

Anonymous Authors[1]

Abstract

Question decomposition, i.e. breaking a complex query into simpler sub-queries whose answers are composed to produce a final answer, is a widely used strategy for improving LLM reasoning, yet it currently lacks a rigorous mathematical foundation. In this paper, we propose operads — mathematical structures that model many-in, one-out operations and compositions thereof — as a natural framework for describing question decomposition. We define the *questions operad* $\mathcal{Q}$, in which operations correspond to question templates and composition corresponds to substitution of sub-answers, and show how QA models can be interpreted as algebras over $\mathcal{Q}$. Beyond reframing existing practice, this operadic perspective points toward new methods — in particular, a notion of *reasoning robustness*, which measures consistency of a QA model's answers across all partial collapses of a question decomposition tree. In experiments across eight models and four multi-hop QA datasets, we find that *operadic consistency* — a scalar instantiation of reasoning robustness — is strongly correlated with accuracy, whereas temperature-based self-consistency is not, suggesting that the operadic notion captures a distinct and useful signal. We argue that operads are the natural mathematical home for question decomposition, and that invariants such as reasoning robustness open new directions for analyzing and improving the reliability of multi-step reasoning.

1. Question decomposition in practice

Large Language Models (LLMs) have demonstrated remarkable reasoning capabilities, particularly when prompted to break problems down into intermediate steps. Such an idea was first popularized in work on chain-of-thought prompting (Wei et al., 2022) and its many subsequent extensions (Yao et al., 2023; Khot et al., 2023; Wang et al., 2023; Yang et al., 2024; Besta et al., 2024). In practice, this process involves transforming a complex query into simpler sub-queries whose answers are sequentially composed to derive a final solution — for instance, answering *How long was it between when the Titanic hit the iceberg and when it sank completely?* by first resolving the timestamps of the collision and the sinking.

Despite its intuitive appeal and widespread adoption, question decomposition currently lacks a rigorous mathematical foundation. This matters: without a formal account of composition structure, it is difficult to define what it means for a decomposition to be correct or well-formed, to reason about how errors propagate through a chain of sub-queries, or to compare decomposition strategies in a principled way — all independently of any particular model implementation.

In this paper, we propose operads as the right mathematical framework for this problem. Operads were introduced by mathematicians to study systems built by iterated substitution — how many-input operations compose coherently into larger operations — and they provide a compact language for tree-shaped composition. In our setting, operations correspond to question templates with blanks, and composites correspond to decompositions into sub-queries. Relatedly, Marcolli et al. (2023) have applied operadic and associahedral structures to model syntactic Merge and the syntax-semantics interface in generative linguistics, further supporting the view that operads are a natural framework for compositional structure in language. Beyond developing the formalism, we also present preliminary empirical evidence that *operadic consistency* — a simple scalar instantiation of reasoning robustness — is a strong across-model predictor of accuracy on multi-hop QA, while a standard temperature-based self-consistency baseline is not.

2. Operads

Operads are structures for organizing collections of "morphisms", i.e. operations that take $k \geq 0$ inputs and return one output. They are closely related to *categories*, which are better-known objects that organize one-in, one-out operations.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Accepted to the 2nd Workshop on Compositional Learning at ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

In this section, we will introduce the reader to operads, and then explain how question decomposition and chain-of-thought can be described in terms of the *questions operad*. We assume no prior knowledge of category theory or operads, and prioritize accessibility. Throughout, the objects we call "operads" are, more precisely, "non-symmetric operads in Set".

2.1. Operads in mathematics

We begin this section with a concise explanation of the notion of an operad. There are few approachable references; we suggest the curious reader consult Markl–Stasheff–Shnider's *Operads in Algebra, Topology and Physics* (Markl et al., 2002), and Loday–Vallette's *Algebraic Operads* (Loday & Vallette, 2012).

Operads model properties of many-in, one-out morphisms. One feature that such morphisms have is that we can compose them. Indeed, consider mappings f and g, which take k and ℓ inputs, respectively. For any $1 \leq i \leq k$, we can form the composition

$$f\Big(\underbrace{-,\ldots,-}_{i-1}, g\Big(\underbrace{-,\ldots,-}_{\ell}\Big), \underbrace{-,\ldots,-}_{k-i}\Big), \tag{1}$$

i.e. the result of feeding the output of g into the i-th input slot of f. The result is a mapping that takes $k+\ell-1$ inputs, which we denote $f \circ_i g$. Iterated compositions are unambiguous — for instance, when we write the expression

$$f(\ldots, g(\ldots, h(\ldots), \ldots), \ldots), \tag{2}$$

the meaning is independent of whether we first compose f with g and then feed in h, or first compose g with h and then feed the result into f.

These observations lead us to the definition of an operad.

Definition 2.1. An *operad* $\mathcal{O}$ is the following data:

- A collection $\mathcal{O}(0), \mathcal{O}(1), \mathcal{O}(2), \ldots$ of sets. (Elements of $\mathcal{O}(k)$ are thought of as arity-k morphisms, i.e. operations with k inputs and one output.)
- An identity morphism $\mathrm{id}_{\mathcal{O}} \in \mathcal{O}(1)$.
- For every $k, \ell \geq 0$ and $1 \leq i \leq k$, a "composition map" $\circ_i \colon \mathcal{O}(k) \times \mathcal{O}(\ell) \to \mathcal{O}(k+\ell-1)$. (When $\mathcal{O}$ is literally a collection of morphisms, and we are given $f \in \mathcal{O}(k)$ and $g \in \mathcal{O}(\ell)$, $f \circ_i g$ is obtained by plugging the output of g into the i-th input slot of f. The result is a $(k+\ell-1)$-ary operation.)

These data must satisfy the following conditions:

- Composition is *associative*. That is, for every $k, \ell, m \geq 0$, $f \in \mathcal{O}(k), g \in \mathcal{O}(\ell), h \in \mathcal{O}(m)$, and $1 \leq i \leq k$, $1 \leq j \leq \ell$, the compositions
$$f \circ_i (g \circ_j h), \quad (f \circ_i g) \circ_{i+j-1} h \tag{3}$$
agree. (Compare (2).)
- Composing with the identity does nothing. That is, $\alpha \circ_i \mathrm{id}_{\mathcal{O}} = \alpha$ and $\mathrm{id}_{\mathcal{O}} \circ_1 \alpha = \alpha$. △

In the rest of the subsection, we describe two operads with connections to classical computer science and formal language theory.

Example 2.2. Given an alphabet Σ, we can form the set Σ^* of finite strings over Σ. We define the *text processing operad over* Σ, denoted $\mathcal{F}_\Sigma$, like so:

- $\mathcal{F}_\Sigma(k)$ is the set of all mappings from $(\Sigma^*)^k$ to Σ^*, i.e. all mappings that take k inputs, all lying in Σ^*, and return one element of Σ^*.
- The identity is the identity map.
- Composition is ordinary composition of mappings, as in (1).

For instance, $\mathcal{F}_\Sigma(0)$ can be identified with Σ^* itself; $\mathcal{F}_\Sigma(1)$ contains the operator sending a string to its all-caps version; $\mathcal{F}_\Sigma(2)$ contains the binary concatenation operator $(x, y) \mapsto xy$; and $\mathcal{F}_\Sigma(3)$ contains ternary operations such as the map sending (x, y, z) to the string obtained by joining x, y, and z with the separator "`,`". Let f and g denote the binary and ternary operators just introduced, respectively. Then $f \circ_1 g$ (i.e., f with g plugged into its first input slot) and $f \circ_2 g$ (i.e., f with g plugged into its second input slot) are distinct elements of $\mathcal{F}_\Sigma(4)$. △

Colored operads *Colored* (or *typed*) operads are like operads, except that each morphism takes inputs of specific *colors*, and returns an output of a specific color (or type). If $\mathcal{O}$ is a colored operad, we denote by $\mathcal{O}(c_1, \ldots, c_k; d)$ the morphisms that take k inputs with colors $c_1, \ldots, c_k$, respectively, and return an output of color d. (A category can be interpreted as a colored operad with only unary operations; for this reason, colored operads are sometimes called *multicategories*.)

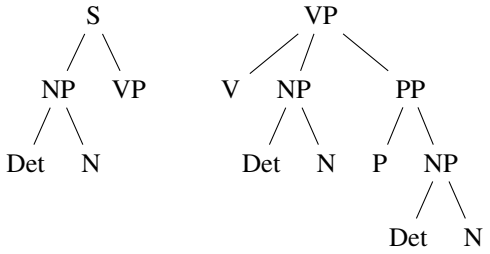
Example 2.3. Suppose that $G = (V, \Sigma, R, S)$ is a context-free grammar, as in §2.1 of Sipser (1996). We define a *derivation tree with non-terminal leaves* for G to be a rooted planar tree T such that:

- Every node is labeled by a non-terminal $A \in V$;
- If an internal node is labeled A, and its children are labeled $B_1, \ldots, B_k$, then $A \to B_1, \ldots, B_k$ must be a production rule in R.

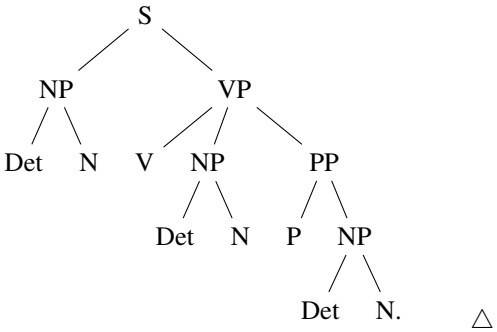
We define D_G to be the colored operad of derivation trees with non-terminal leaves for G, with colors V. That is:

- Given $B_1, \ldots, B_k \in V$ and $A \in V$, $D_G(B_1, \ldots, B_k; A)$ consists of the derivation trees for G whose k leaves have labels $B_1, \ldots, B_k$ and whose root has label A.
- Composition in D_G is given by grafting the root of one derivation tree to a leaf of another — a common operation in context-free parsing — subject to the requirement that the labels at the two nodes being identified must match.

For instance, suppose that G is a context-free grammar modeling English, with non-terminals {S, NP, VP, PP, Det, N, V, P} (standing for "sentence", "noun phrase", etc.). Here are two derivation trees, called T_1 and T_2, which define elements of $D_G(\text{Det}, \text{N}, \text{VP}; \text{S})$ and $D_G(\text{V}, \text{Det}, \text{N}, \text{P}, \text{Det}, \text{N}; \text{VP})$, respectively:



We can form the composition $T_1 \circ_3 T_2$, which grafts the trees at their unique VP node to yield an element of $D_G(\text{Det}, \text{N}, \text{V}, \text{Det}, \text{N}, \text{P}, \text{Det}, \text{N}; \text{S})$:

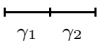


△

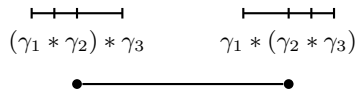
Why operads are helpful. The appeal of operads is that they allow us to separate the abstract forms of composition from any particular interpretation of those forms. For example, Examples 2.2-2.3 talk about very different objects (i.e., k-ary string processing functions vs. derivation trees in formal grammars), yet operads package both as instances of the same underlying idea: an operation with multiple inputs and one output, together with a notion of plugging one operation into an input slot of another.

Operads were originally introduced by Peter May in algebraic topology (May, 1972), where one motivating problem was to understand the algebraic structure of loopspaces. The basic idea of a loop is simple: fix a point x_0 in a space X, and consider continuous paths $\gamma\colon [0,1] \to X$ that start at x_0 (i.e. $\gamma(0) = x_0$), trace out a route through X, and return to x_0 at the end of their interval (i.e. $\gamma(1) = x_0$). The loopspace ΩX is the collection of all such loops.

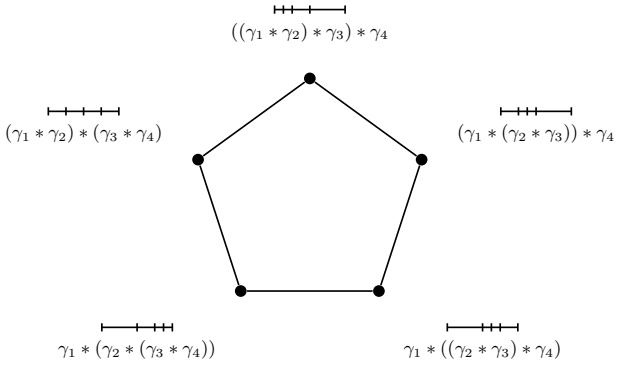
Loopspaces are compositional because one can traverse one loop γ_1 and then a second γ_2, resulting in the *concatenation* $\gamma_1 * \gamma_2$. We can depict this concatenation as the unit interval $[0,1]$, with a tick mark indicating the transition from γ_1 to γ_2:



Concatenation, however, is not associative. Indeed, $(\gamma_1 * \gamma_2) * \gamma_3$ and $\gamma_1 * (\gamma_2 * \gamma_3)$ traverse the same loops, but do so with different speeds and are therefore distinct as loops. These concatenations are nevertheless deformable into one another by reparametrizing time, which corresponds to moving the tick marks on the interval. There are thus two ways to concatenate three loops, related by a single deformation; we organize them into the endpoints of an interval, the *associahedron* $\mathcal{K}(3)$:



For four loops, there are five parenthesizations. This time, the deformations that relate these concatenations are organized into a pentagon, the associahedron $\mathcal{K}(4)$:



Unlike the purely combinatorial operad D_G, the associahedra are polytopes — topological spaces whose cells record not merely that parenthesizations are related, but the continuous geometry of how they deform into one another. One might instead simply identify all parenthesizations as homotopy-equivalent and work with the quotient; but this would discard the finer structure that concatenation naturally carries.

May's insight was that these coherence data are precisely

the kind of structure operads encode. In modern terms, loopspaces are naturally algebras over the associahedra operad, and May's Recognition Principle says, roughly, that under suitable hypotheses this operadic structure characterizes loopspaces. This illustrates the "operadic mantra":

Objects with extra structure should be understood as algebras over an appropriate operad.

This example also connects to the grammar and question-decomposition examples below. The associahedra organize different parenthesizations of a composite expression; derivation trees similarly record different histories for composing constituents into a string; and question decompositions record different histories for composing subquestions into a final query. In all three cases, the compositional history itself carries information. Operads provide a language for studying that history, rather than only the final output.

We have two purposes to bring the language of operads to bear in the context of question decomposition and machine learning more generally. First, we believe that operads are simply the natural formal structure for question decomposition, and therefore that their introduction in this context is innately worthwhile. Second, reformulating question decompositions and QA models in operadic terms can lead to new results and tools — for instance, a new notion of "reasoning robustness" for QA models (see §2.3.2). Even in the well-studied context of context-free grammars, we describe in Remark 2.6 how the operadic point of view can point to new tools.

2.1.1. Algebras over operads

Operads do not necessarily have to literally be collections of operations (for instance, consider D_G from Example 2.3). A realization of an operad in terms of actual operations is called an *algebra*.

Definition 2.4. Suppose that $\mathcal{O}$ is a colored operad. An *algebra $\mathcal{A}$ over $\mathcal{O}$* is:

- For every color c, a set $\mathcal{A}(c)$.
- For every $f \in \mathcal{O}(c_1, \ldots, c_k; d)$, a map

$$\varphi_f \colon \mathcal{A}(c_1) \times \cdots \times \mathcal{A}(c_k) \to \mathcal{A}(d). \tag{4}$$

We require that $\varphi_{f \circ_i g}$ is the same operation as inserting the output of φ_g into the i-th input of φ_f; this condition is also referred to as "associativity". △

For instance, Σ^* is naturally an algebra over $\mathcal{F}_\Sigma$: for each $f \in \mathcal{F}_\Sigma(k)$, the corresponding algebra operation is simply the map $f : (\Sigma^*)^k \to \Sigma^*$. (In fact, this is the only $\mathcal{F}_\Sigma$-algebra.) We describe another example of an algebra, this time over the colored operad D_G from Example 2.3, below.

Example 2.5. Given a context-free grammar, we can define an algebra L_G over the operad D_G of derivation trees for G, like so:

- For $A \in V$, $L_G(A)$ is the language generated by A, i.e. the language of the subgrammar with A as the start symbol.
- If T is a derivation tree for G, with root label A and leaf labels $B_1, \ldots, B_k$, then it defines a map

$$\varphi_T \colon L(B_1) \times \cdots \times L(B_k) \to L(A) \tag{5}$$

by concatenation. For instance, the operation associated to the derivation tree described near the end of Example 2.3 could take as inputs {"The", "dog", "saw", "the", "cat", "in", "the", "park"} and return the sentence "The dog saw the cat in the park". △

The content of a context-free grammar can therefore be repackaged as the operad D_G and its algebra L_G. Importantly, L_G is only one possible algebra over D_G. Indeed, besides the string yield algebra, one can define algebras for computing derivation probabilities, best-parse computations, or derivation counts. This connects to classical work on weighted and semiring parsing (Goodman, 1999; Nederhof, 2003), where different computations are obtained by varying the value algebra while keeping the underlying derivational machinery fixed. The operadic viewpoint complements this line of work by treating the grafting structure of derivations as a first-class algebraic object. This is useful not only for computing values over derivations, but also for defining more complex algebraic constructions that capture properties of the derivational system itself, such as the ambiguity algebra introduced below and, later, our notion of reasoning robustness for question decomposition.

Remark 2.6. At least as early as Chomsky–Schützenberg's foundational work (Chomsky & Schützenberger, 1959), *ambiguity* has been regarded as a central notion in the theory of context-free grammars and languages. One treatment can be found in §5.4 of Hopcroft et al. (2001), where the authors define a CFG G to be unambiguous if each string has at most one derivation tree in G. This definition is binary and has no internal algebraic structure.

Our reformulation of a CFG as an operad D_G and an algebra L_G enables us to define a new avatar of ambiguity, called the *ambiguity algebra* $\ker \varphi$. This is a canonical object that recovers the usual definition of ambiguity as the condition $\ker \varphi \neq 0$, while additionally encoding the compositional structure of ambiguity.

To define the ambiguity algebra, we first linearize D_G and L_G to form $D_G^{\mathbb{R}}$ and $L_G^{\mathbb{R}}$. That is, we define $D_G^{\mathbb{R}}(B_1, \ldots, B_k; A)$ to consist of formal $\mathbb{R}$-linear combinations of elements of $D_G(B_1, \ldots, B_k; A)$, and $L_G^{\mathbb{R}}(A)$ similarly. $L_G^{\mathbb{R}}$ is then an algebra over $D_G^{\mathbb{R}}$. We can define a

second $D_G^{\mathbb{R}}$-algebra, $F_G^{\mathbb{R}}$, which is the linearization of the D_G-algebra F_G consisting of those complete derivation trees whose leaves are all labeled by terminals, and where the D_G-action is defined by grafting. There is a forgetful map, $\varphi\colon F_G^{\mathbb{R}} \to L_G^{\mathbb{R}}$, which sends a derivation tree to its yield. We now define the ambiguity algebra to be the kernel $\ker\varphi$, i.e. those elements of $F_G^{\mathbb{R}}$ that are sent to zero by φ. △

2.2. Operads for question decomposition

We will now apply the language of operads to the context of question decomposition. Our proposal takes particular inspiration from variants of chain-of-thought prompting, particularly the decomposed prompting approach of Khot et al. (2023) and uncertainty of thoughts approach of Hu et al. (2024) that decompose input problems to sets of subquestions. We first define an operad, $\mathcal{Q}$, in terms of questions and how they can be de- and re-composed. Then we will explain how QA models can be interpreted as algebras over $\mathcal{Q}$.

Definition 2.7. The *questions operad* $\mathcal{Q}$ consists of the following data:

- For every $k \geq 0$, $\mathcal{Q}(k)$ is the set of questions with k blanks.
- For every $k, \ell \geq 0$ and $1 \leq i \leq k$, we define a composition operation

$$\circ_i\colon \mathcal{Q}(k) \times \mathcal{Q}(\ell) \to \mathcal{Q}(k+\ell-1) \tag{6}$$

  that plugs the output of the second question into the i-th blank of the first.
- There is a distinguished "identity question" $\mathrm{id}_{\mathcal{Q}} \in \mathcal{Q}(1)$, defined to be "Return the following, verbatim: `[A1]`.'' △

It is natural to construct $\mathcal{Q}$ as a colored operad. This involves a choice of answer types, defined by the user. For instance, we could set

$$\mathrm{Color}(\mathcal{Q}) := \{\text{time}, \text{duration}, \text{place}, \text{person}, \text{other}\}.$$

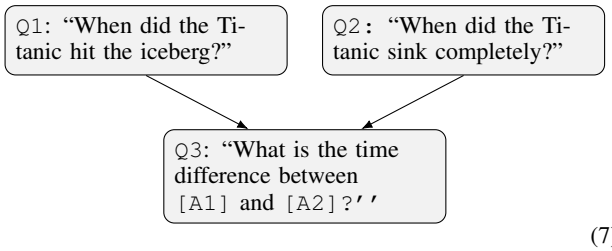
Example 2.8. Consider the following questions:

$$\texttt{Q1} := \text{"When did World War 2 end?"} \in \mathcal{Q}(;\text{time}),$$
$$\texttt{Q2} := \text{"Who was President at } - \text{"?} \in \mathcal{Q}(\text{time};\text{person}),$$
$$\texttt{Q3} := \text{"Who was } - \text{'s wife?"} \in \mathcal{Q}(\text{person};\text{person}).$$

The composition $\texttt{Q3} \circ_1 \texttt{Q2} \circ_1 \texttt{Q1}$ is "Who was First Lady when World War 2 ended?" (or a semantic equivalent). In this case, the associativity of $\mathcal{Q}$ says that this composition is independent of whether we first compose `Q3` and `Q2` or `Q2` and `Q1`. △

Remark 2.9. Definition 2.7 should be understood at an abstract level: the composition $\circ_i$ is formal substitution of one typed question template into the i-th blank of another. Operationally, however, one may wish to render such substitutions as fluent natural-language questions, and that surface-realization step will depend on a model. Accordingly, associativity is best understood up to semantic equivalence. △

We can interpret question decomposition in terms of composition in $\mathcal{Q}$. For instance, consider the question `Q`, "How long was it between when the Titanic hit the iceberg and when it sank completely?" We can decompose `Q` into the following tree of questions:



(7)

We can reinterpret this in terms of the questions operad:

- `Q` is an element of $\mathcal{Q}(;\text{duration})$. `Q1` and `Q2` lie in $\mathcal{Q}(;\text{time})$, and `Q3` lies in $\mathcal{Q}(\text{time},\text{time};\text{duration})$.
- The fact that (7) is a legitimate decomposition of `Q` is witnessed by the fact that `Q1`, `Q2`, and `Q3` compose to `Q` — that is, $\texttt{Q} = \texttt{Q3} \circ_1 \texttt{Q1} \circ_1 \texttt{Q2}$.

Any tree of questions such as this one can be interpreted in a similar way.

2.2.1. ML MODELS AS ALGEBRAS OVER OPERADS

Now that we have interpreted question decomposition in terms of the questions operad, we can interpret a QA model as an algebra over $\mathcal{Q}$. In the following definition, we use the term *value* to mean something that can be used to fill in a blank, or something that can occur as the output of a question.

Definition 2.10. Suppose that m is a general-purpose question answering model, that takes a question q and returns an answer v. We can use m to define a $\mathcal{Q}$-algebra, $\mathcal{V}_m$, in the following way:

- As a set, $\mathcal{V}_m$ consists of all possible values.
- Given a question $q \in \mathcal{Q}(k)$ with k blanks, along with k values $v_1, \ldots, v_k$, we can produce a value v' by the following procedure:

- Fill in the k blanks in q with $v_1, \ldots, v_k$, then process in order to form a question q'.
- Define v' to be the output of m on q'. △

2.3. A notion of reasoning robustness resulting from our operadic interpretation

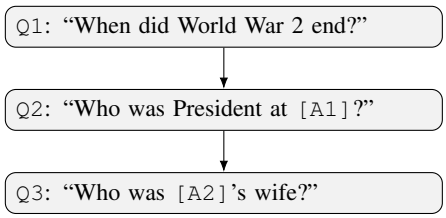
Operads do not only provide a new language for describing question decomposition — they also point the way toward new methods. In this subsection, we will define a notion of *reasoning robustness*, given a QA model m and a tree of questions T.

2.3.1. Trees of questions and partial collapses thereof

Fix an instantiation of the questions operad $\mathcal{Q}$. By a "tree of questions" (or "ToQ") T, we mean the following:

- A rooted tree T. By convention, (1) we orient all trees toward the root, (2) each leaf has a single incoming edge, and (3) the root has a single outgoing edge.
- For every edge e in T, a color $t_e \in \mathrm{Color}(\mathcal{Q})$.
- For every vertex $v \in T$, a question $q \in \mathcal{Q}(t_{e_1}, \ldots, t_{e_k}; t_{e'})$, where $e_1, \ldots, e_k$ are the incoming edges of v and e' is the outgoing edge of v.

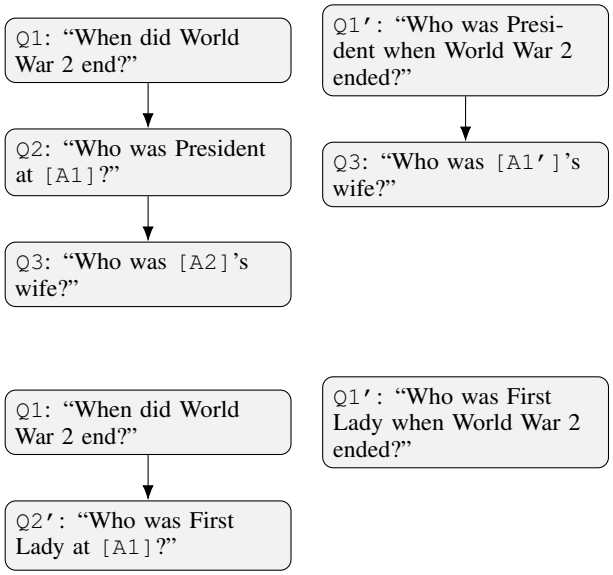
One example is the ToQ in (7). Another is formed by the questions in Ex. 2.8, which we use as a running example in this subsection:



(8)

Any ToQ T has the property that the questions can be composed to form a question with no blanks. We call this the *total collapse* of T. On the other hand, we can form a *partial collapse* by composing some, but not all, of the questions involved. Indeed, given a ToQ T and a choice, for every edge, of whether to compose along that edge, we can produce a partial collapse of T. The total number of partial collapses of T is $2^{\#T-1}$, where $\#T$ is the number of vertices of T.

Example 2.11. There are $4 = 2^{3-1}$ partial collapses of the ToQ in (8):



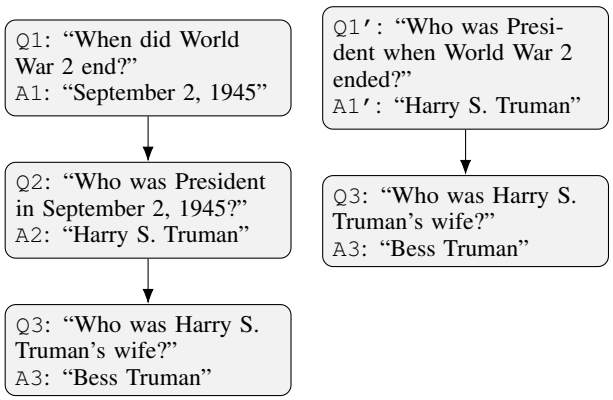
△

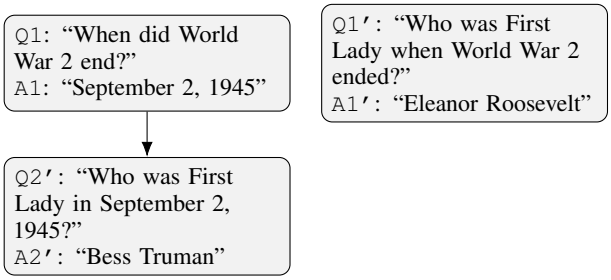
2.3.2. Reasoning robustness

Suppose that T is a tree of questions and m is a QA model. We can use m to answer the questions in T: we start at the leaves and work our way toward the root, finally finishing by producing an answer to the root question. If $\mathcal{V}_m$ is associative on the nose, then the final answer produced by m on T agrees with m's final answer on the total collapse of T. Moreover, if T_1' and T_2' are two partial collapses of T, then the final answers produced by m on T_1' and T_2' must agree. This leads to our definition of *reasoning robustness*.

Definition 2.12. We say that m *is robust on* T if, for any partial collapses T_1', T_2' of T, the final answers produced by m when executed on T_1' and T_2' agree. △

Example 2.13. Consider $m :=$ Llama 3 8B Instruct. When we execute m on the partial collapses of the ToQ T in (8), it produces the following answers:





Non-robustness is witnessed by the answer to collapse 4 disagreeing with the answers to the others. △

2.4. Empirical evidence: operadic consistency predicts accuracy

Reasoning robustness (§2.3.2) is a predicate: a model either is or is not robust on a given tree of questions. To evaluate this quantitatively, we introduce *operadic consistency*, a scalar instantiation that, for a two-step decomposition tree, compares just the two extreme partial collapses. We now provide preliminary empirical evidence that operadic consistency is a meaningful proxy for QA accuracy and outperforms self-consistency (Wang et al., 2023).

Setup. We evaluate eight instruction-tuned models spanning roughly 4B to 671B parameters (Llama 3 8B, Gemma 3n 4B, Qwen2.5 7B, LiquidAI 24B, Llama 3.3 70B, Qwen3 235B, DeepSeek V3.1, Cogito 671B) on four multi-hop QA datasets (HotpotQA, MuSiQue, StrategyQA, DROP; 403–625 questions each), using human-authored two-step decomposition trees: native decompositions for MuSiQue and StrategyQA, and Break QDMR-high-level annotations (Wolfson et al., 2020) for HotpotQA and DROP. For each (model, question) pair, *operadic consistency* is the indicator that the model's greedy direct answer and its greedy decomposed answer (answer step 1, substitute into step 2, answer) agree at token F1 ≥ 0.5; *self-consistency* is the mean pairwise token F1 agreement over K samples at temperature 0.7, for K=3 and K=10. Accuracy is token F1 ≥ 0.5 between the greedy direct answer and the gold answer.

Results. Figure 1 plots each consistency metric against accuracy across the eight models, per dataset. Operadic consistency is strongly correlated with accuracy on all four datasets (Pearson r between 0.70 and 0.98; $p \leq 0.003$ on three, $p = 0.053$ on DROP), while self-consistency shows no significant correlation at either K=3 or K=10 (all $|r| \leq 0.37$, all $p \geq 0.36$). That increasing K does not help suggests the limitation is not sample size but the nature of the signal.

Limitations. These results are correlational and across-model, not within-model: they do not yet show that operadic consistency can be used as a per-question filter or selection signal, nor do they compare against stronger baselines such as chain-of-thought self-consistency (Wang et al., 2023).

3. Further directions

The operadic framework introduced in this paper opens several directions for future work, both theoretical and experimental.

From across-model correlation to per-question selection. The empirical results of §2.4 establish operadic consistency as an across-model predictor of accuracy. A natural next step is to ask whether it is also useful *within* a single model, on a per-question basis — that is, whether questions on which a model is operadically consistent are answered correctly at higher rates than those on which it is not. A positive result here would turn operadic consistency from a diagnostic into a tool, usable as an inference-time confidence signal for deployed QA systems, with the attractive property of requiring no ground-truth labels. Further directions include comparing operadic consistency to stronger baselines such as chain-of-thought self-consistency (Wang et al., 2023) and exploring deeper decomposition trees.

Extracting operadic structure from chain-of-thought. A natural extension of the framework is to apply it to thinking models, whose extended reasoning traces make the implicit decomposition structure explicit. In ongoing work, we develop methods for extracting a formal chain of questions (CoQ) from a model's reasoning trace, and use this to test whether the model's own stated reasoning is self-consistent in the operadic sense — that is, whether re-deriving the intermediate steps independently yields the same final answer. Preliminary results on Qwen3-235B (thinking mode) across MuSiQue, StrategyQA, DROP, and GSM8K show that consistency remains a strong predictor of accuracy even in this setting.

Cohomological invariants. Algebras over operads admit a notion of *cohomology*, and we believe that the cohomology of the $\mathcal{Q}$-algebra associated to a QA model may carry meaningful information about the structure of a model's inconsistencies — distinguishing, for instance, between correctable inconsistencies and more fundamental failures of multi-step reasoning.

Other algebras over $\mathcal{Q}$. The operadic perspective suggests that question decompositions need not be interpreted only by their final answers. They may also be interpreted by likelihoods, confidence scores, evidence bundles, costs, latent representations, or robustness profiles, each giving a different algebra over the same underlying compositional structure.

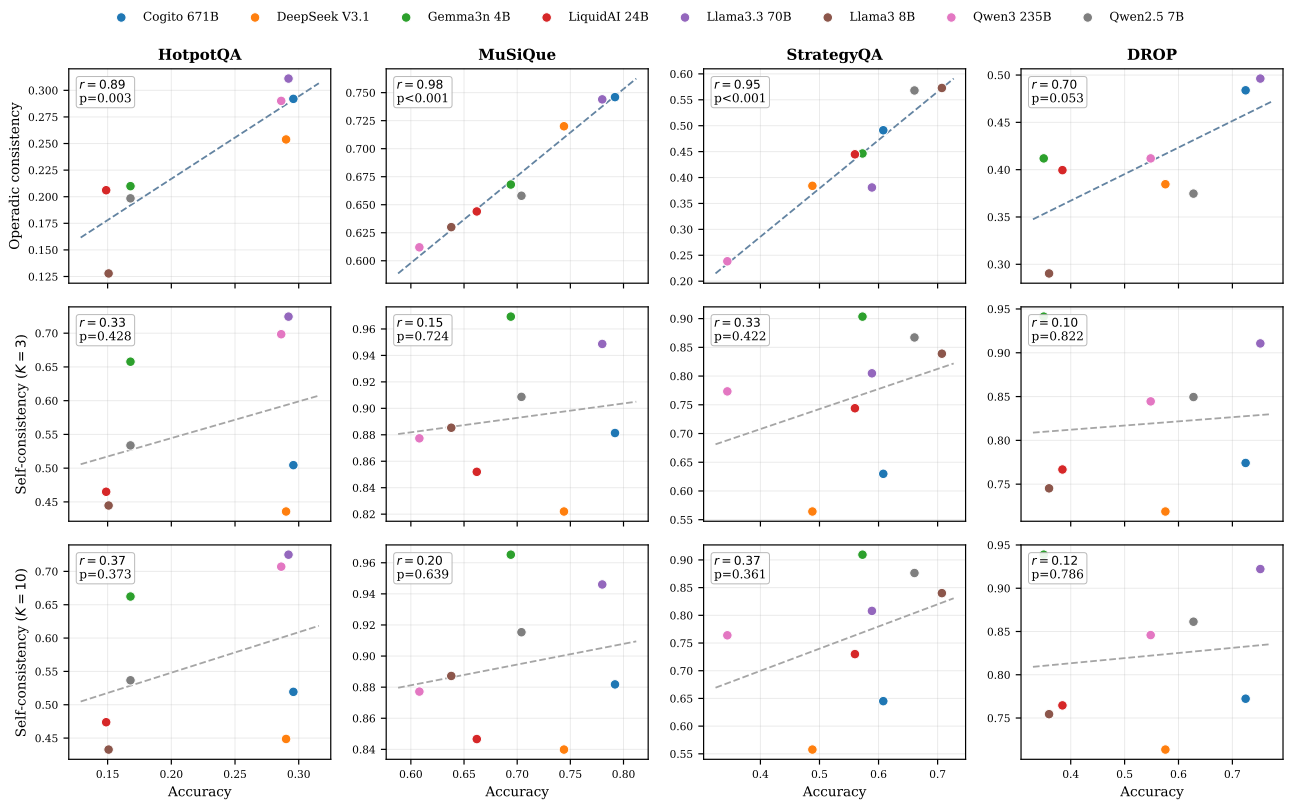


Figure 1. Consistency vs. accuracy across eight models and four datasets. **Top row:** operadic consistency is strongly correlated with accuracy on all four datasets. **Middle and bottom rows:** self-consistency (Wang et al., 2023) (mean pairwise token F1 agreement over K temperature-0.7 samples) shows no significant correlation at $K{=}3$ or $K{=}10$. Each point is one model; trend line and Pearson r with two-sided p are shown per subplot.

Impact Statement

This paper develops a mathematical framework and preliminary empirical evaluation for question decomposition in language models. We foresee no direct negative societal impacts; reasoning robustness could in principle be used to audit deployed QA systems.

References

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Chomsky, N. and Schützenberger, M. P. The algebraic theory of context-free languages. In *Studies in Logic and the Foundations of Mathematics*, volume 26, pp. 118–161. Elsevier, 1959.

Goodman, J. Semiring parsing. *Computational Linguistics*, 25(4):573–606, 1999.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. Introduction to automata theory, languages, and computation. *ACM Sigact News*, 32(1):60–65, 2001.

Hu, Z., Liu, C., Feng, X., Zhao, Y., Ng, S.-K., Luu, A. T., He, J., Koh, P. W., and Hooi, B. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in large language models. *arXiv preprint arXiv:2402.03271*, 2024.

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., and Sabharwal, A. Decomposed prompting: A modular approach for solving complex tasks. *Proceedings of ICLR*, 2023.

Loday, J.-L. and Vallette, B. *Algebraic operads*, volume 346. Springer Science & Business Media, 2012.

Marcolli, M., Berwick, R. C., and Chomsky, N. Syntax-semantics interface: an algebraic model. *arXiv preprint arXiv:2311.06189*, 2023.

Markl, M., Shnider, S., and Stasheff, J. Operads in algebra, topology and physics. *Mathematical surveys and monographs*, 96, 2002.

May, J. *The Geometry of Iterated Loop Spaces*, volume 271 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, Heidelberg, 1972. ISBN 978-3-540-06012-3. doi: 10.1007/BFb0067491.

Nederhof, M.-J. Weighted deductive parsing and knuth's algorithm. *Computational Linguistics*, 29(1):135–143, 2003.

Sipser, M. *Introduction to the Theory of Computation*. Computer Science Series. PWS Publishing Company, 1996. ISBN 9780534952501. URL https://books.google.com/books?id=t1I_AQAAIAAJ.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *Proceedings of ICLR*, 2023.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wolfson, T., Geva, M., Gupta, A., Gardner, M., Goldberg, Y., Deutch, D., and Berant, J. Break it down: A question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8:183–198, 2020.

Yang, L., Yu, Z., Zhang, T., Cao, S., Xu, M., Zhang, W., Gonzalez, J. E., and Cui, B. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37: 113519–113544, 2024.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.