The Reasoning Mechanism of AI Neural Networks

A Clear, Neuronal-Level Account of Language, Reasoning, and Other Semantic Transformations

Jianyu Duan* and Mingjun Duan[1]

[1]AEQ AI Research Institute, New Zealand
*jyduan.aeq@gmail.com

Abstract

How neural networks actually reason remains a mystery. This paper identifies the core mechanism: language understanding and reasoning are definite computations over internal semantic structure, realized as conditional semantic transformations performed by cognitive neural operators. These operators recognize concepts, relations, and roles, compare them, suppress invalid cases, and transform valid semantic structures into new ones. We make this mechanism explicit through a parameter-level circuit that maps "C belongs to A; all A have B" to "C has B," The complete word-to-conclusion pipeline is specified at the level of individual weight parameters and passes all 16,680 exact simulation tests. This mechanism is further verified experimentally: trained models form the theory-predicted cognitive-reasoning process stages and functionally equivalent operator topologies, or variants thereof. This provides the first explicit, neuron-level, parameter-interpretable realization of natural-language reasoning—distinct from Boolean truth-value circuits, which cannot express genuine semantics, and from the qualitative circuit descriptions typical of large-model interpretability work. The same neural-operator mechanism also offers a cogent account of emergence, scaling laws, generalization, and hallucination, and points toward neuron-level operator interpretability, hallucination control, AI-safety tooling, neural editing, and theory-guided model design. In our experiments, this mechanism shifts model architecture design from empirical search toward mechanistic calculation: the required cognitive operators and reasoning chains can be used to theoretically derive model width, neuron count, and network depth, and to locate immature operators, layers, and local structures during training for targeted optimization. Small-model experiments show that this mechanism-guided approach achieves more efficient training, lower compute cost, and faster inference. More broadly, it reduces cognition from the level of symbolic rules to neuron-level computation, opening the black box at the level of mechanism.

Author Contributions

Jianyu Duan proposed a series of concepts on semantic transformation and formed the *Meaning Activity Theory of Thinking*, a systematic theory of thinking based on semantic formation and semantic transformation [6–10]; he completed a series of theorems on recognition, classification, and perception, and provided neural-operator circuit structures—every explicit parameter clearly interpretable—that fully realize natural-language reasoning as inferential semantic transformation. Mingjun Duan, with an interdisciplinary background in cognitive science and artificial intelligence (natural language processing) from Carnegie Mellon University, took part in model training and theoretical analysis, and proposed studying interpretability at the neuronal level, enabling a concrete, material account of the cognitive processes addressed here.

1. Introduction

A neural network that answers a reasoning question correctly tells us *what* it did, not *how*. Despite the fluency of today's language models, we cannot point to the computation inside a network that constitutes an act of reasoning—and without that, it remains open whether these systems reason at all or only match patterns. A Microsoft study of GPT-4 called uncovering the mechanisms behind such emergent abilities an urgent task [2], and a large interpretability literature has grown around the question without reaching the level of concrete computation [16, 19, 20, 22]. This paper sets out to answer it: we give an explicit, mechanism-level account of how a neural network realizes language and reasoning—not as outward behavior, but as a definite, inspectable computation over internal semantic structure.

Two existing styles of explanation fall short. Boolean truth-table circuits express truth values but not the semantics of language or the transformations that act on it; recent qualitative circuit descriptions of large models trace suggestive pathways [1, 15] but do not specify the mechanism at the level of individual weights, biases, and activations. What is missing is an account that is at once *semantic*—about meaning, not merely truth value—and *concrete*—realized by explicit parameters.

Our thesis is that such an account exists and is simple in form: *language reasoning is the conditional semantic transformation of one internal semantic structure into another, realized through cognitive neural operators.* Building on Jianyu Duan's *Meaning Activity Theory of Thinking* [6–10], in which cognition is the formation and transformation of internal semantic structure driven by mapping operators, a cognitive neural operator is a neural structure—weights, biases, activations, and connection paths—that performs a cognitively interpretable function: concept formation, recognition, classification, comparison, gating, suppression, semantic transfer, or concretization, among others. Because an operator is defined by its computation rather than its substrate, the same functional problem recurs across symbolic, artificial-neural, and biological systems (Figure 1); we develop its neural realization here.

A single familiar example makes the mechanism concrete. The categorical syllogism

$$[\,C,\ \textit{belongs-to},\ A\,] \ +\ [\,\textit{all},\ A,\ \textit{have},\ B\,] \ \longrightarrow\ [\,C,\ \textit{has},\ B\,] \tag{1}$$

is not a truth-value operation but a semantic transformation: the premises are recognized, the two occurrences of the *middle term* A (the shared category that links the premises) are checked for consistency, the entities C and B are preserved, and the conclusion is released only when the relation (*belongs-to*), the quantifier (the counting word *all*), the predicate (the property word *have*), and the match of A all hold. The syllogism is used because it is compact and inspectable; the theory is not derived from it but from the broader principle that thinking transforms meaning-bearing forms. Throughout we keep formalism light, giving exact parameters only where they make an operator visible.

The significance is mechanistic: rendering reasoning as an explicit operator circuit, rather than a behavioral or attention-level description, opens the black box at the level of mechanism, and reframes the debate over whether models "reason" or "do statistics" [3]—statistical training can form operators that encode logical regularity, and here that mechanism is made explicit.

Our contributions are mechanism-level rather than the construction of one circuit:

- an account of language reasoning as the conditional semantic transformation of internal semantic structure, with recognition and classification stated as theorems at the level of neuronal operator structure (Sections 2–4);
- parameter-level neural mechanisms for the operators—recognition, classification, concept generation, comparison, conditional gating, suppression, semantic transfer, and output concretization—including the two basic gate forms, bias-filling and negative-weight blocking, that conditional reasoning requires (Sections 3–6);
- a complete, parameter-level realization of a basic language-reasoning transformation, verified exactly over all valid and invalid cases (Sections 7–8);
- controlled computational evidence that freely trained dense networks and a standard Transformer realize the same operator functions as topological variants, establishing the operator account as an interpretability framework for trained models (Section 10; full details in a companion experiment report); and
- a neuron-level operator account of large-model phenomena—emergence, scaling, generalization, and hallucination—a reading of independent interpretability findings as correspondences to the predicted operators, and the resulting significance for interpretability, AI safety, and system design (Sections 11–13).

This is a mechanism-level account in the literal sense: we establish the operators by explicit construction and exact verification, corroborate them in freely trained networks, and trace their consequences for large models, with the controlled experiments detailed in a companion experiment report.

2. Reasoning as Conditional Semantic Transformation: Why It Requires Cognitive Neural Operators

We first fix the conceptual foundation on which the operator mechanism rests. For the purpose of this paper, we focus on semantic formation and semantic transformation. The

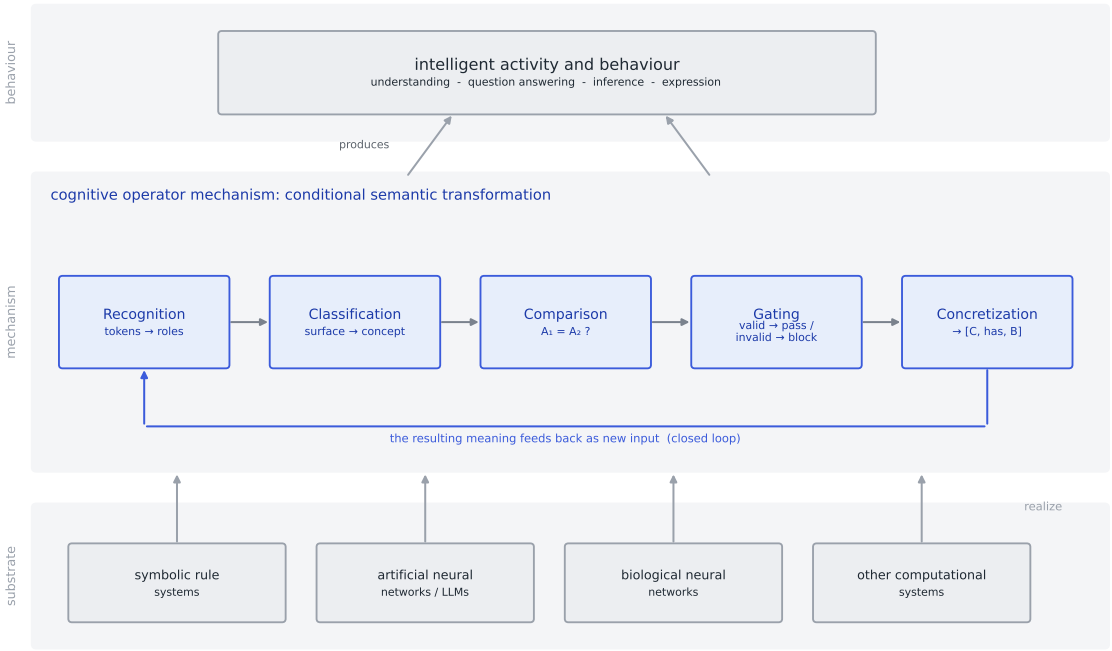


Figure 1: The cognitive neural operator mechanism, situated between intelligent activity and behavior above and its possible substrate realizations below. The operator level is a closed loop of conditional semantic transformation—recognition, classification, comparison, gating, concretization, and others—whose result feeds back as new input. The stages shown are a schematic illustration of one possible flow rather than an exhaustive or fixed sequence, each standing for one example of its operator family. Symbolic-rule systems, artificial neural networks and large language models, biological neural networks, and other computational systems differ in substrate but face the same functional problem; this paper develops the artificial and biological neural realization.

broader Meaning Activity Theory also includes representation, interpretation, memory, expression, and control systems. In its earlier operator language, an object or event can be mapped into a meaning-bearing representation by a representation operator K, transformed by a sequence of operators, and interpreted or expressed back through an output-side operator K^{-1}. We develop the neural realization of the central transformation stage while keeping these wider functions in view.

2.1. Semantic formation

Semantic formation is the process by which inputs become internal semantic representations. Inputs may be words, symbols, images, or sounds; inside a neural network they are represented as numerical, vectorial, or distributed activation structures. Both biological and artificial networks convert directly perceivable forms of expression into quantitatively represented internal structures that carry implicit semantic content—an electrochemical signal in biological tissue, a vector or embedding in an artificial network. In language models, tokens are mapped by an embedding layer into vectors distributed across hidden units; in either neural setting, once information enters the network it is held in a quantifiable form on which operators can act. Semantic formation is not a purely external-input mapping: in the broader theory it is state-dependent, arising from the interaction between input information and the system's current internal state. Meanings are also carried by *expressive forms*: language is one such form, but symbols, images, sounds, actions, and internal vector representations can also function as expressive forms once they carry semantic content. Recent findings that human-like object-concept representations and structured concept geometries emerge in trained models are consistent with this view of internal semantic content [5, 14].

2.2. Semantic transformation

Semantic transformation is the process by which one internal semantic structure becomes another. Attribute inheritance, question answering, paraphrase, coreference resolution, formal inference, predicate substitution, and conditional-conclusion generation are all instances. Language reasoning is a specific, structured form of semantic transformation: the syllogism of Eq. (1) carries the premise structure into the conclusion structure under stated conditions. Language is our focus here, but semantic transformation is not limited to language; it can operate over any internally represented expressive form.

2.3. Why a controlled semantic transformation requires functional stages

A controlled semantic transformation cannot be carried out by a single undifferentiated step; it requires a small set of functional stages, each of which is necessary in the sense that removing it breaks the transformation. Without internal representation, the system can only manipulate surface signals and has no meaning to transform. Without recognition and classification, it cannot tell, for a particular thing, which relation, quantifier, or condition is present, nor recognize the regularities common to things of the same kind. Without role binding, it cannot tell which symbol is the member C, which is the class A, and which is the property B. Without relational comparison, it cannot check the relations between terms,

including whether the two occurrences of the middle term are the same or otherwise related. Without conditional gating and suppression, it would release a conclusion even when the conditions fail. Without semantic transformation proper, no new conclusion structure is produced from the premises. And without concretization, the internal result cannot be expressed as a context-appropriate sentence, symbol, action, or answer. These stages are therefore not a particular engineering choice but the functional decomposition that any controlled semantic transformation must realize; the neural operators below are their realization as neural computational structures.

2.4. Cognitive neural operators

A *cognitive neural operator* is a neural computation structure that realizes a cognitively interpretable function such as recognition, classification, comparison, conditional gating, conditional suppression, semantic transfer, or output concretization. It is realized by neurons, weights, biases, activation functions, and connection paths, and is typically composite. Put plainly, it is a small group of neurons wired to carry out one identifiable cognitive step of reasoning—recognizing a feature, comparing two values, or opening and closing a path—rather than merely computing an arbitrary number. Two properties matter for what follows. First, these operators are not ordinary mathematical operators but *conditional* mapping operators: the operator is triggered only when specified conditions on the input features hold. Second, a neural operator is defined by the spatial connectivity structure of its constituent neurons and its computation, not by its material substrate; from the operator-function point of view, biological and artificial neural networks can be treated as functionally comparable, even though their material substrate and learning dynamics differ. In the language of the earlier theory, cognitive neural operators are neural realizations of transformation operators acting on expressive forms, while routing and conditional gating correspond to part of the path-selection and control function. In an explicitly specified network an operator may appear as a clean module or pathway; in a freely trained network the same function may appear as a distributed, split, cross-shared, compressed, or stretched topological variant. We state this once here and return to it in Section 10; the body of this paper works first with the explicit form.

3. The ReLU Neuron and Its Two Gate Forms: The Building Block of Operator Computation

3.1. The neuron as a thresholded conditional unit

The basic unit computes

$$a \;=\; \mathrm{ReLU}\Big(\textstyle\sum_i w_i x_i + b\Big), \qquad \mathrm{ReLU}(z) = \max(0,\, z), \tag{2}$$

where each input x_i is multiplied by a weight w_i, the products are summed, a bias b is added, and the result is passed through the activation function ReLU. For readers outside the field, ReLU is the simplest nonlinearity used in modern neural networks: it leaves a

positive value unchanged and replaces any zero or negative value with exactly zero—a one-sided "valve." We read the unit functionally rather than numerically: a *positive* weight lets an input support the unit, a *negative* weight lets an input suppress it, the *bias* sets a threshold, and the ReLU nonlinearity passes the net input when it is above threshold and blocks it (outputs zero) otherwise. Smooth activations such as GELU, SiLU, and SwiGLU realize the same thresholded behavior with soft rather than sharp boundaries; we therefore speak of ReLU-equivalent activation, and the mechanism does not depend on the exact choice.

3.2. Conditional pass-through

The combination of weighted summation and a thresholded nonlinearity implements a *conditional pass-through*: when a required condition is absent the unit outputs zero, and when it is present the payload passes through, faithfully or scaled. This is the elementary act from which recognition and gating are built.

3.3. Weights, thresholds, and cancellation

In this reading, weights are not merely numerical coefficients but functional relations among features. A positive weight lets a feature contribute support, a negative weight lets it contribute opposition, and the size of the weight expresses how strongly that feature matters for crossing the threshold. Several weak supports can add to open a path; opposing inputs can cancel; and a single sufficiently large negative input can block an otherwise supported output. Thus comparison, recognition, and conditional control are all expressed by the same threshold arithmetic: the net input either crosses the activation boundary or it does not.

3.4. The two gate forms: opening on a condition, closing on an error

A neuron can act as a *gate*: a small set of condition features decides whether the remaining payload features pass. Two complementary implementations suffice. Both are the same neuron equation (2), but they realize the gate in opposite ways—one by *filling* a deep negative bias, the other by *suppressing* with a deep negative weight.

Bias-filling gate. In plain terms, the payload is held switched off by a deep negative bias until the required condition signal arrives and fills exactly that gap, switching the path on; the gate responds to whatever feature encodes the condition—not to any particular keyword. Formally, give the payload a large negative bias $-B$, where B is an upper bound on the payload's own contribution, $B \geq \max_x \sum_{j\in\mathcal{P}} w_j x_j$, and let a condition feature x_c enter with a weight w_c chosen so that $w_c x_c = B$ exactly when the condition is present:

$$a \;=\; \mathrm{ReLU}\Big(w_c x_c + \textstyle\sum_{j\in\mathcal{P}} w_j x_j - B \Big). \tag{3}$$

Then

$$a = \begin{cases} \sum_{j\in\mathcal{P}} w_j x_j > 0, & \text{condition present } (w_c x_c = B):\ \text{payload passes unchanged,} \\ 0, & \text{condition absent } (x_c = 0):\ \text{net input } \leq 0,\ \text{payload blocked.} \end{cases}$$

Negative-weight blocking gate. In plain terms, the path stays open until a forbidden or mismatched feature appears and forces the output down to zero. Formally, keep the bias moderate and attach any mismatching or forbidden feature through a large negative weight $-M$, with d the error evidence and $\mathcal{S}$ the supporting inputs:

$$a \;=\; \mathrm{ReLU}\Big(\; \textstyle\sum_{i\in\mathcal{S}} w_i x_i - M\, d + b \Big), \qquad M \gg 0. \tag{4}$$

Then

$$a = \begin{cases} \mathrm{ReLU}\big(\textstyle\sum_{i\in\mathcal{S}} w_i x_i + b\big), & \text{no error } (d=0): \\ & \text{supporting signal passes,} \\ 0, & \text{error present } (d>0): \\ & -Md \text{ drives the input below 0.} \end{cases} \tag{5}$$

The two gates are duals: the bias-filling gate *opens* a path when a required condition arrives, the blocking gate *closes* a path when a forbidden feature arrives. They can be subsumed under one neuron equation, but they are genuinely different implementation strategies, and a reasoning circuit typically uses both.

3.5. Why two gate forms suffice for conditional reasoning

Given ReLU-equivalent thresholded computation, the requirement to preserve a valid payload, and the requirement to block invalid paths, these two gate forms are the natural and sufficient mechanisms for conditional reasoning: one opens an output on a satisfied condition, the other closes it on a detected error, and together they realize "pass if and only if the conditions hold." They explain how a network performs conditional reasoning without any explicit symbolic `if`-statement, the condition being carried entirely by weights, biases, and the activation boundary. We do not claim these are the only possible gates, only that they are sufficient and natural for conditional reasoning; how far the operator forms are necessary in general we leave open here.

4. Recognition and Classification Operators: Selecting Meaning by Interval

Because feature recognition is the precondition for semantic transformation, we begin with it and state the results as theorems at the level of neuronal operator structure.

4.1. Interval recognition: passing a feature only within its range

Consider a one-dimensional feature value X and a target interval $[n, n+c]$ with $c \geq 0$. Using a lower-boundary unit, an upper-boundary unit, a payload path, and a sufficiently large suppression strength M, a small ReLU network outputs the input when it lies inside the interval and zero otherwise (Figure 2, inclusion). Reversing the boundary signs yields the complementary operator, which passes the input outside the interval and blocks it inside.

Recognition Theorem 1 (Feature recognition and preservation)**.** *A* ReLU *neural operator can recognize a scalar feature within a specified interval* $[n, n+c]$ *and preserve its value,*

outputting X when $n \le X \le n + c$ and 0 otherwise; an analogous operator realizes the complementary (exclusion) recognition.

Explicitly, two boundary units detect the upper and lower violations and sum them into out-of-range evidence d, which a blocking gate (Eq. (4)) then uses to suppress the payload outside the interval:

$$\begin{aligned} d &= \mathrm{ReLU}\big(X - (n + c)\big) + \mathrm{ReLU}\big(n - X\big), \\ R_{\mathrm{in}}(X) &= \mathrm{ReLU}\big(X - M\,d\big) = \begin{cases} X, & n \le X \le n + c, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

Inside the interval both detectors are zero, so $d = 0$ and the payload X passes unchanged; outside, one detector is positive, so $-Md$ drives the output to zero. The complementary exclusion operator reverses this, passing X outside the interval and blocking it inside:

$$R_{\mathrm{ex}}(X) = \begin{cases} 0, & n \le X \le n + c, \\ X, & \text{otherwise.} \end{cases}$$

Unlike a single separating hyperplane, the operator carves out a bounded region and can return the original feature value for downstream use; its boundaries may be sharp or, with smooth activation, graded, which is what later admits fuzzy and modal recognition.

4.2. Classification and category recognition

A small modification maps a whole interval to a single category value rather than preserving the input. Using the same out-of-range evidence d as in Eq. (6) to gate a constant m,

$$\begin{aligned} K(X) &= \mathrm{ReLU}\big(m - M\,d\big) = \begin{cases} m, & n \le X \le n + c, \\ 0, & \text{otherwise,} \end{cases} \\ \bar{K}(X) &= \begin{cases} m_2, & X < n \text{ or } X > n + c, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{7}$$

so the operator emits the shared category value m for *any* input in the interval, independent of its exact value, and the complementary operator emits a second category value m_2 for inputs outside it (Figure 3). Such classification operators represent the category rather than the original input, and the same principle extends from a scalar feature to an N-dimensional feature vector by combining one-dimensional recognition of the components with positional and structural constraints. This yields the second theorem.

Recognition Theorem 2 (Category recognition from feature vectors)**.** *A* ReLU *neural operator can recognize a category from any N-dimensional real-valued feature vector within the representable range, mapping every vector in the admissible region to a shared category value; the one-dimensional interval classification of a single feature is the special case $N = 1$.*

4.3. Recognition and classification: why they underlie the language operators

Recognition and classification operators are therefore the basis of semantic selectivity: a network can recognize a feature and preserve it, exclude it, or map it into a concept class. Any feature representable within the numerical and precision range of the network can in this way be recognized, classified by interval, and individually identified—the foundation on which the later language operators rest. For the worked diagrams below we use simple positive scalar values, but the theorem is not limited to that convention. Finite intervals, complementary unbounded intervals, discrete values, and continuous ranges are all handled by the same boundary topology; zero, negative values, decimals, and other numerical ranges require only adjusted weights, biases, or an equivalent shifted representation.

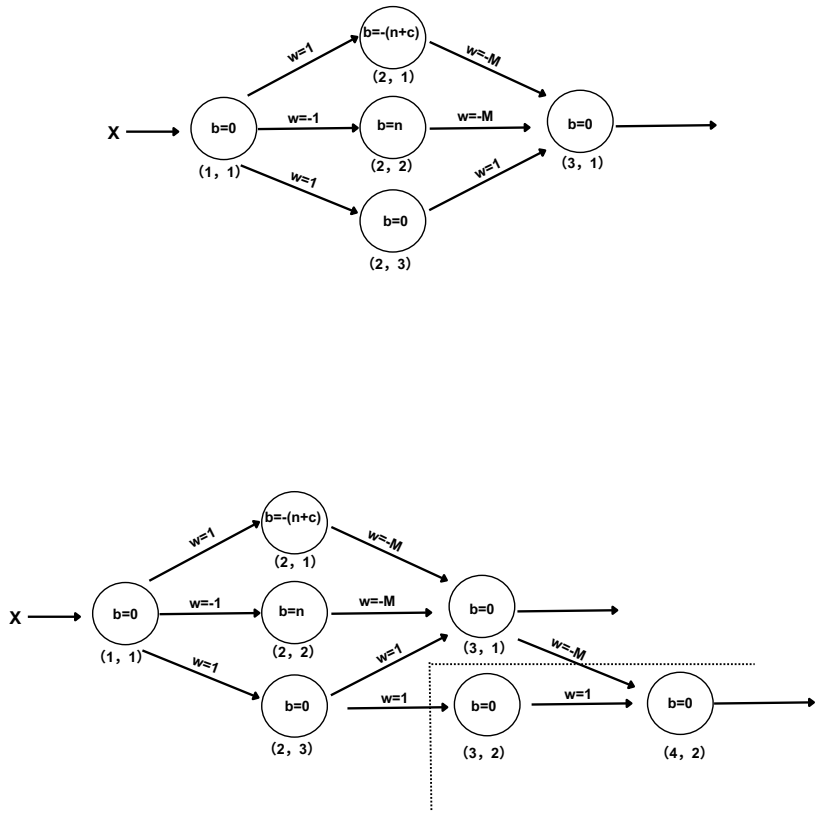


Figure 2: Recognition operators. Top: inclusion recognition, output = X inside $[n, n + c]$ and 0 outside. Bottom: the complementary exclusion operator. Each circle is a ReLU unit; edge labels are weights and node labels are biases; M is a large suppression strength.

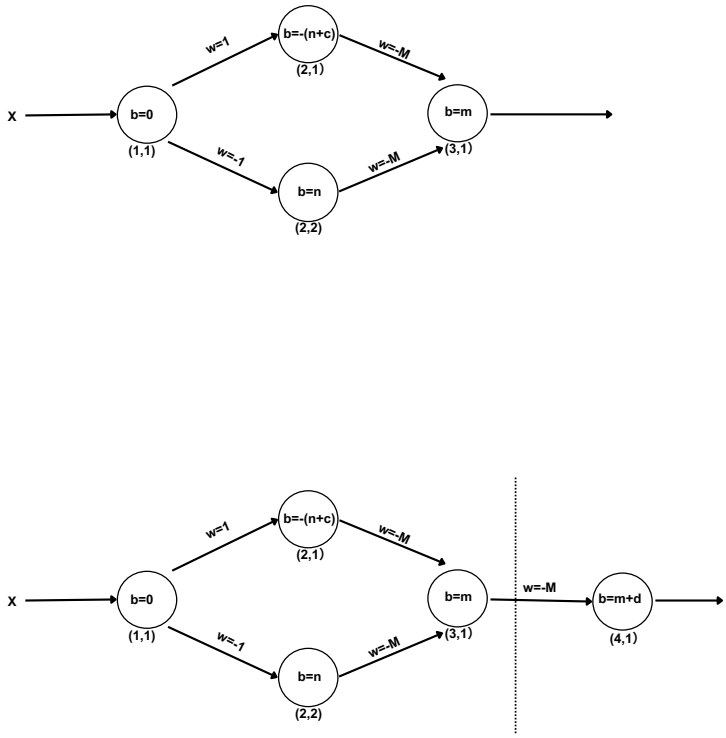


Figure 3: Classification operators. Top: inputs in $[n, n + c]$ are mapped to a single category value m. Bottom: the complementary operator maps inputs outside the interval to $m + d$.

5. Concept Generation and Normalization: From Surface Tokens to Reasoning-Ready Concepts

Language reasoning operates over concepts, not raw surface tokens, so the surface input must first be turned into concept-level features.

5.1. Compound concept generation

Lower-level tokens combine into a higher-level semantic unit: *belongs* and *to* form the relation concept *belongs-to*, just as *hot* and *dog* form *hot dog*. A small operator with appropriate weights and a bias threshold fires only when the constituent features co-occur,

$$c \;=\; \mathrm{ReLU}(\, w_1 t_1 + w_2 t_2 - \theta \,), \tag{8}$$

where t_1, t_2 are the constituent features (e.g. *belongs*, *to*) and the threshold θ is set so that $c > 0$ only when both are present; a single constituent leaves the sum below θ and gives $c = 0$. The compound value is then normalized to a canonical concept by a classification operator (Figure 4). In a full language network many such operators can coexist in early layers, so that multiple candidate word, phrase, and relation concepts are generated in parallel and later selected or suppressed by recognition and contextual constraints.

5.2. Expression normalization

Once each multi-token relation has been composed into a compound value by Eq. (8)—*belongs+to* into one *belongs-to* value, *is+part+of* into another—the different surface forms of the same relation—*belongs to*, *is in*, *is part of*, *is among*—are mapped, by classification operators of the kind in Theorem 2, to one shared membership concept. Normalization lets the reasoning mechanism use a single circuit for a relation rather than a separate circuit for every surface phrasing, and it is the point at which the network becomes robust to paraphrase.

5.3. From surface expressions to operator-ready structures

The bridge from natural language to the reasoning circuit has several levels. At the sentence level, expressions such as "C belongs to A; all A have B; therefore C has B" and "C is in A, and everything in A has B" can express the same semantic transformation. At the premise level, the first relation may appear as *belongs to*, *is part of*, *is among*, or *is in*. At the feature level, these phrases are first assigned *distinct* raw values that fall within a single recognition interval—for example 107–110 inside $[101, 120]$—and an interval-recognition operator (Section 4) then collapses that whole interval to *one* canonical value (here 7) used by the reasoning operator. Normalization is thus a two-step move: distinct surface forms take distinct raw values in a shared interval, and interval recognition maps the interval to a single concept value. The same applies to the universal quantifier and predicate: *all*, *every*, or *each* fall in the quantifier interval $[201, 220]$ and normalize to 1, while *have*, *has*, *possess*, or related expressions fall in $[301, 320]$ and normalize to 3. These upstream outputs feed the corresponding input positions of the reasoning circuit, such as the relation input

$(1, 2)$, the quantifier input $(1, 4)$, and the predicate input $(1, 6)$ in Figure 5. Thus the circuit is not manually supplied with an idealized logical form; it receives a normalized semantic structure produced by earlier recognition and classification operators. In Figure 5, the normalized relation, quantifier, and predicate inputs should therefore be read as the outputs of upstream recognition/classification operators: when the relevant class is recognized, the operator outputs its canonical value, such as 7 for *belongs-to*, 1 for *all*, and 3 for *have*; when the class is absent or invalid, the corresponding output is 0. Thus invalid relation, quantifier, or predicate features do not enter the reasoning circuit as arbitrary nonzero values.

5.4. Role in reasoning: what concept generation hands to the later operators

Concept generation and normalization supply the concept-level features—relation, quantifier, predicate, and the entity payloads—that the comparison, gating, and transformation operators then act upon. Concept generation compresses surface variants into shared abstract features; output concretization performs the reverse mapping, expanding a shared internal result into a context-appropriate expression. The predicate *have*, for example, may be normalized to a single internal category for reasoning, or in some designs its surface form may be preserved by a recognition operator and later reused. Likewise, the payload entities C and B can be carried as stable internal values while their final linguistic expression is chosen by context. In the earlier operator notation, this concretization corresponds to the output-side or interpretation-side mapping K^{-1}: it maps an internal semantic structure back into an expressive form, such as a sentence, symbol, action, or interpreted object relation. Without this stage, reasoning would have to operate directly on surface strings; with it, reasoning operates on a normalized semantic structure and then returns to language through concretization.

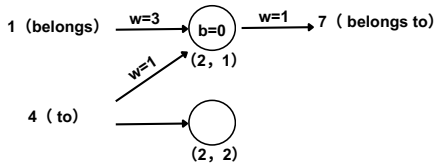
6. Comparison and Conditional Control: Enforcing Consistency and Gating the Conclusion

6.1. The comparison operator

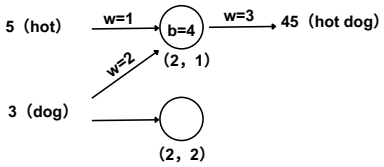
The simplest comparison is equality. Whether two concept features A_1 and A_2 agree is decided by forming their difference and recognizing whether it lies in a small tolerance interval around zero—a direct application of Theorem 1 to the signal $A_1 - A_2$. Concretely, the mismatch evidence is the absolute difference, computed by two ReLU units,

$$d \;=\; \mathrm{ReLU}(A_1 - A_2) + \mathrm{ReLU}(A_2 - A_1) \;=\; |A_1 - A_2|, \qquad \text{match} \iff d \le \varepsilon, \tag{9}$$

which is zero when $A_1 = A_2$ and positive otherwise. A match ($d = 0$) opens a path; a mismatch ($d > 0$) produces error evidence that the blocking gate of Eq. (4) uses to close the output. Equivalent realizations include similarity scores, explicit match/mismatch signals, and attention-style alignment.



Part A: example of a Concept Vector Generation Operator for "belongs to".



Part B: example of a Concept Vector Generation Operator for "hot dog".

Figure 4: Concept generation operators. Constituent tokens, such as *belongs+ to* or *hot+dog*, are combined through weighted paths and a bias threshold into a single compound-concept feature value. The numerical values shown are illustrative coding values, not lexical meanings themselves. After compound formation, distinct surface forms of one relation can be normalized by classification operators to a shared concept value.

6.2. Comparison beyond equality: the general relation operator

Equality is only the simplest case. The same operator family decides greater-than and less-than (the sign of a difference), similarity (whether a distance is small or an inner product large), membership (whether a class score crosses a threshold), and spatial or predicate consistency—each reduced to whether a scalar relation feature lies in an admissible interval. Variable binding (keeping one symbol's value consistent wherever it recurs), predicate substitution (replacing a predicate's argument when a condition holds), coreference resolution (deciding that two expressions refer to the same entity), and context consistency all rest on this general comparison.

6.3. Role binding and conditional control: keeping each concept in its place

Reasoning requires that each concept occupy the right *role*—which symbol is the member (C), which is the class (A), and which is the property (B), and likewise for the subject, predicate, and scope of a sentence. These role assignments are carried by positional and structural constraints in the wiring (realized by position encoding or by the intrinsic order of the circuit), so that, for instance, the two occurrences of the middle term are the ones compared. In the explicit circuit below this role assignment is built into the input positions; in a general language system it is implemented by position-encoding neural operators, a related operator family designed separately and not expanded here. Conditional control then combines the recognized conditions and the comparison result through the two gates: valid configurations open the transformation, while an invalid relation, quantifier, predicate, or a mismatched variable each routes error evidence into the suppression path and closes the

output.

7. A Parameter-Level Syllogism Circuit: Assembling the Operators into Reasoning

We now assemble the operators into an explicit circuit that realizes the transformation of Eq. (1), shown in Figure 5 with every neuron's bias, every connection's weight, and the ReLU activation made explicit.

7.1. Inputs, conditions, and target

The circuit in Figure 5 operates on concept-level features, not on raw token strings. The left front end first recognizes and normalizes word and phrase tokens into the first-column inputs of the reasoning circuit. Thus the circuit receives the normalized semantic structure

$$[\, C, 7, A_1, 1, A_2, 3, B \,],$$

where 7 denotes the normalized *belongs-to* relation, 1 denotes the normalized universal quantifier *all*, and 3 denotes the normalized predicate *have*. These values are outputs of upstream recognition/classification operators, not arbitrary raw token values. If a required relation, quantifier, or predicate class is not recognized, the corresponding normalized input is 0. The target output is $[\, C, 3, B \,]$, i.e. $[\, C, \text{has}, B \,]$, if and only if the relation belongs to the membership class, the quantifier belongs to the universal class, the predicate belongs to the have/has class, and the two middle terms match, $A_1 = A_2$. Otherwise the circuit outputs the zero vector. The payload entities C and B are preserved and released only when these conditions hold.

7.2. Numerical assumptions for the worked realization

The specific parameter values in Figure 5 are chosen for a transparent demonstration. The normalized condition inputs take their canonical values when their classes are recognized—7 for *belongs-to*, 1 for *all*, and 3 for *have*—and take 0 when the corresponding class is absent or invalid. The payload values C, A_1, A_2, and B are taken as positive values not exceeding 10^6, and unequal middle-term values are assumed to differ by at least one unit, so that a fixed large negative weight can suppress the output when $A_1 \neq A_2$. These are demonstration choices, not theoretical restrictions. Other numerical ranges can be handled by rescaling, shifting, or using equivalent recognition structures.

7.3. Operator composition: assembling the circuit with the two gate forms

The circuit implements the transformation by composing the general operators developed above. The concrete ReLU arithmetic below is not a special-purpose trick of this circuit, but one parameter-level realization of the general cognitive-neural-operator mechanism; the computation can then be followed directly. First, recognized condition features cancel large negative biases so that the relevant payload values can pass. For example, the *belongs-to* value 7, with weight $10^6/7$, contributes exactly 10^6, canceling a bias of -10^6 and allowing C

and A_1 to pass. Likewise, the *all* value 1, with weight 10^6, cancels the corresponding -10^6 bias and allows A_2 to pass. Second, the two middle terms are compared by computing the two ReLU differences

$$\mathrm{ReLU}(A_1 - A_2), \qquad \mathrm{ReLU}(A_2 - A_1).$$

Together these give zero mismatch evidence when $A_1 = A_2$, and positive mismatch evidence when $A_1 \neq A_2$. Third, any nonzero mismatch is sent through large negative weights, such as -10^6 or -100, to suppress the downstream conclusion. Thus the final output path remains open only when the required relation, quantifier, predicate, and middle-term equality conditions all hold. In operator terms, this is the composition of recognition/classification, conditional pass-through, comparison, invalid-path suppression, semantic transfer, and output concretization. When the conditions hold, the predicate value 3 and the payload values C and B are released as $[\,C, 3, B\,]$; otherwise the output is suppressed to the zero vector.

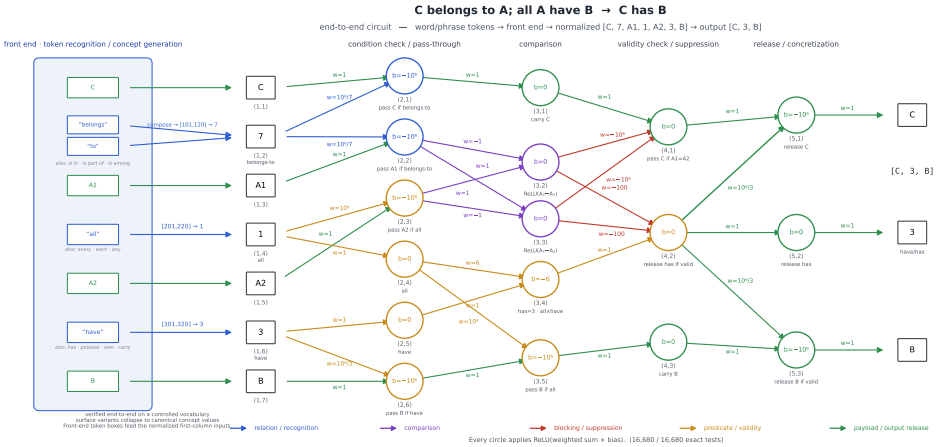


Figure 5: Parameter-level realization of the basic language-reasoning transformation $[\,C, \text{belongs-to}, A\,] + [\,\text{all}, A, \text{have}, B\,] \rightarrow [\,C, \text{has}, B\,]$. Each circle is a ReLU unit, with its bias shown inside and its neuron index shown below; each edge carries the displayed weight. The left front end performs token recognition, concept generation, and classification, mapping word and phrase tokens to the normalized first-column inputs of the reasoning circuit: *belongs-to* is represented by 7, *all* by 1, and *have* by 3. If the corresponding class is absent or invalid, the normalized input is 0. The reasoning circuit then compares A_1 and A_2. If $A_1 = A_2$ and the required relation, quantifier, and predicate conditions are present, the circuit releases $[\,C, 3, B\,]$, i.e. $[\,C, \text{has}, B\,]$; otherwise the large negative suppression paths drive the output to $[\,0, 0, 0\,]$. Functionally, the circuit composes the general cognitive neural operators introduced above: recognition and classification, concept generation and normalization, comparison, conditional pass-through, invalid-path suppression, semantic transfer, and output concretization. The full pipeline is verified end to end on a controlled vocabulary in Section 10.

8. Tracing and Verifying the Circuit: Does It Actually Reason?

8.1. How the circuit reasons

The computation can be followed directly from Figure 5. The left front end supplies normalized inputs: *belongs-to* $= 7$, *all* $= 1$, *have* $= 3$, and invalid or absent condition classes are represented as 0. Take a valid example with

$$C = 2, \quad A_1 = 5, \quad A_2 = 5, \quad B = 4.$$

The condition values cancel the corresponding large negative biases, so C, A_1, A_2, and B can pass. The comparison units give

$$d = \mathrm{ReLU}(A_1 - A_2) + \mathrm{ReLU}(A_2 - A_1) = |5 - 5| = 0.$$

Because $d = 0$, the negative suppression paths are inactive, and the final output is

$$[\,2, 3, 4\,],$$

that is, $[\,2, \mathrm{has}, 4\,]$.

Now change only the second middle term to $A_2 = 6$, keeping $C = 2$, $A_1 = 5$, and $B = 4$. Then

$$d = |5 - 6| = 1.$$

This nonzero mismatch evidence enters the downstream suppression paths through large negative weights. For example, the final property release is suppressed as

$$\mathrm{ReLU}(4 - 10^6 \cdot 1) = 0.$$

The circuit therefore outputs

$$[\,0, 0, 0\,].$$

Thus the circuit releases the conclusion exactly when the required relation, quantifier, predicate, and middle-term equality conditions hold.

8.2. Verification: exact simulation over all valid and invalid cases

We verified the circuit by simulation. Evaluated over the full set of valid and invalid inputs—across the admissible relations, quantifiers, predicates, and middle-term pairings—the circuit reproduces the target transformation exactly: it releases the correct conclusion for every valid premise pair and suppresses the output to zero for every violation of relation, quantifier, predicate, or middle-term match. The reasoning is thus not asserted but checked at the level of the specified parameters.

9. From the Syllogism to General Language Reasoning: Composing the Operators

9.1. Composition: how basic operators combine into complex reasoning

The worked circuit is one instance of a general pattern. The same operator families—recognition, classification, concept generation, comparison, gating, suppression, transfer,

and concretization—combine in series, in parallel, nested, and conditionally coordinated, to realize attribute inheritance, conditional reasoning, predicate substitution, coreference resolution, question answering, paraphrase, active–passive transformation, context completion, multi-step relation reasoning, and, with graded boundaries, fuzzy and modal inference. Complex language reasoning is, on this account, the coordinated combination of basic cognitive neural operators.

At the sentence level, the chain is more concrete: perceptual or token inputs first generate word, phrase, and relation features; recognition and classification select the relevant concepts; position-encoding operators assign each concept its role; pattern matching selects among learned transformation patterns; comparison operators test equality, similarity, or mismatch; conditional gates perform the semantic transformation; and concretization maps the internal result back into expression. Language reasoning can in essence be seen as a composition of primitive transformations on symbol strings—addition, deletion, replacement, shifting, and permutation—all implemented by conditionally opening and closing these paths. Expressions such as *most*, *likely*, or *usually* use the same operator skeleton with graded recognition and threshold margins rather than hard all-or-none boundaries, which is why fuzzy and modal reasoning fall under the same mechanism. Residual or skip pathways can preserve earlier payload information so that later operators can reuse it during output generation.

10. Empirical Evidence: The Neural-Operator Reasoning Mechanism Verified in Simulation Experiments and Trained Networks

The preceding sections give an explicit operator construction. The computational question is whether this construction is only a hand-specified artifact, or whether the same functional process stages and their operator topology also appear when networks are trained from data. We summarize the evidence here; the full controlled experiments are reported in a companion experiment report.

10.1. Exact parameter-level verification from tokens to conclusion: 16,680 tests passed

The hand-specified circuit above, simulated over all valid and invalid inputs, reproduces the syllogistic natural-language reasoning exactly. Crucially, the verified system is not the reasoning core alone: it includes the token-recognition and concept-generation front end, so the test runs end to end from word and phrase tokens, passing all 16,680 exact tests in a controlled vocabulary. These results establish that the proposed neural operators and reasoning circuits are not merely analogies; at the parameter level they compute the specified semantic transformation.

10.2. Free-trained dense networks, verified across runs

In a second line of evidence, dense networks are trained without masks and without intermediate supervision: only the final conclusion vector is supervised. Across several representative embedding-dimensional settings, including 128D, these networks solve the syllogism

task, releasing valid conclusions and suppressing invalid cases. Post-hoc probes show a stable pattern. The validity gate is cleanly and linearly recoverable in late layers, while the interior conditions—relation recognition, quantifier recognition, predicate recognition, and A_1/A_2 matching—are often present only in nonlinear or distributed form. Increasing dimension does not automatically make these interior operators cleanly separable. Read at the parameter level, the comparison operator is in fact a clean antisymmetric $A_1 - A_2$ difference unit in the low-dimensional networks, becoming distributed at higher width, so the low linear decodability of the binary match reflects reorganization rather than the absence of the operator. This supports the distinction between *operator function* and *operator topology*: trained networks can realize the same functional stages without preserving the clean one-operator-one-path structure of the explicit construction. We distinguish several such topological variants of one operator: it may be *distributed* (spread smoothly across many units and layers), *split* (its sub-conditions divided among separate sub-circuits), *cross-shared* (occupying the same units as other operators, which then jointly carry several roles), *compressed* (folded into fewer dimensions than its explicit form), or *stretched* (unfolded across more layers than the minimal construction). These are functionally equivalent realizations of the same operator, not failures to implement it—a vocabulary that, unlike the loose notion of "entanglement," names *how* the operator is reorganized.

10.3. Transformer operator topology, consistent with the theory-predicted circuit process stages

A standard attention-plus-ReLU Transformer trained on the same task provides a third check. In the trained model the same operator topology reappears as a distributed realization of the recognition/gating form $\mathrm{ReLU}(X_{\mathrm{valid}} - \sum_i M_i d_i)$: attention supplies routing and error-evidence transport, ReLU feed-forward units supply the thresholded gating and invalid-path suppression, and output readout paths carry payload concretization. The structure is stable across random seeds, consistent with the functionally equivalent topology the theory predicts, and residual-level causal interventions show that the validity and comparison decision is carried causally by the residual decision state—swapping that state between a valid and an invalid run flips the conclusion between release and suppression. By a cardinality criterion (a class is genuinely compressed only when its members outnumber the units that control it), the relation, quantifier, and predicate conditions read as recognition rather than clean category compression, whereas the compound validity class is a genuine category operator: a few shared units, one of them opening for every valid case and closing for every invalid one. Explicit circuits make the mechanism readable; trained networks distribute, split, cross-share, compress, or stretch to achieve equivalent operator functions.

10.4. Further controlled studies: widely used activations, the range of reasoning types, and generalization

The same account has been checked well beyond the single syllogism circuit and its freely trained counterparts; we state here only what was done and what was found, with numbers in the companion experiment report.

- **Masked construction.** A masked, theory-initialized full-chain network trained end

to end reaches strict accuracy 1.0 with each operator stage on its own nodes, and its trained parameters can be read back out as the staged operator chain; the upstream word/phrase-to-circuit pipeline passes all 16,680 exact tests.

- **Basic operators under free training.** Dense networks trained on final outputs alone form recognition, condition-opening, and mismatch-blocking units, and removing the bias/threshold degree of freedom breaks interval recognition—so the threshold term is functionally load-bearing, not incidental.
- **Activation invariance.** The interval-recognition operator is learned not only under ReLU but equivalently under GELU, SiLU, SwiGLU, and attention-style edges, supporting a dependence on ReLU-equivalent thresholded computation rather than on one specific activation.
- **Range of reasoning types.** Beyond the categorical syllogism, the account is corroborated across distinct kinds of reasoning—propositional and predicate rules (modus ponens, conjunction, disjunction, disjunctive syllogism, existential introduction), nested and combined inferences, fuzzy and modal reasoning, and analogical transfer; for representative cases the same validity-gate and comparison structure is recovered by probe and confirmed by causal ablation.
- **Generalization from structure.** When the operator is reused across steps, the network generalizes compositionally to unseen chain lengths and templates, where an unstructured dense network of comparable size does not; under ablation, only theory-structured networks form the clean staged chain.
- **A discriminating test.** Varying how many conditions the task conjoins shows that only the conditions actually required suppress the output, irrelevant conditions leave it unchanged, and a shared AND-gate forms only when the task is conjunctive—a prediction that could have failed but did not, indicating the operators track the real task conditions rather than spurious correlates.

10.5. Universality of structure, diversity of realization

Necessity here does not mean a unique circuit. The universality is twofold. The process stages—concept generation, recognition and classification, comparison and gating, role binding and transformation—recur in any network that performs this reasoning, and the core operators they require either share consistent topological structures or recur as functionally equivalent topological variants. The diversity is that this structure can be distributed across layers, made coincident or shared, lifted into higher dimensions, or spread over several paths; the two gate forms are interchangeable, and because composite operators embed gates, their combinations multiply. The interpretive point matters for real models. Overstate the diversity and a large model looks like a lawless black box; overstate the fixed wiring and every model is expected to show one identical diagram. The accurate reading is that a trained model need not reproduce the explicit circuit above, yet must contain the same process regularities and core operators—or their functionally equivalent topological variants and their compositions—that the circuit makes visible. Language reasoning is in this sense governed by a set of underlying operator laws, and the complexity of real models is those

laws unfolding in high-dimensional, distributed form and, through composition, nesting, and abstraction, evolving into higher-order, more complex operator structures that form intricate neural circuits.

10.6. Operator search targets, edit points, and a biological bridge

For interpretability, the operator account gives concrete search targets: recognition features, comparison paths, validity gates, suppression pathways, semantic-transfer paths, and output-concretization routes. For neural editing, these same structures identify possible intervention points: a reasoning failure may arise from relation recognition, variable comparison, gate opening, invalid-path suppression, payload transfer, or output concretization, rather than from the final distribution alone. The same operator language also connects to biological neural systems at a functional level. Excitatory and inhibitory inputs can be read as positive and negative weights, threshold crossing as bias, and supra/sub-threshold firing as one-sided gating; findings of shared representational structure between language models and human neural activity, neural dynamics aligned with real-time language processing, synaptic motifs for decision computation, and brain-like specialization in deep networks are consistent with this operator-level view [4, 11–13]. The claim is not material identity between artificial and biological networks, but a shared level of description in terms of conditional neural operators.

11. Explaining Large-Model Phenomena: Emergence, Scaling, Generalization, and Hallucination

The mechanism established above is exhibited on a compact task, but its components—recognition, concept generation, comparison, the two gates, suppression, transfer, and concretization—are general. We now state what the operator account predicts for large-scale models. The account gives concrete, testable structure to several otherwise puzzling phenomena. At this level the meaning processing of a language model abstracts to a staged flow

$$\text{input symbols} \to \mathsf{C} \to \mathsf{R} \to \mathsf{D} \to \mathsf{G} \to \mathsf{T} \to \mathsf{E}, \tag{10}$$

where C is concept generation, R recognition and classification, D comparison, G conditional gating, T semantic transfer, and E concretization, with auxiliary operators (position encoding, memory retrieval, error correction, safety suppression, multimodal alignment, and others) acting alongside. Each phenomenon below is read through the same internal quantities: operator counts, recognition-interval widths, gate thresholds, blocking strengths, and tolerances.

Emergence. A capability emerges as a critical transition: its various operators form and close into a complete loop, and the internal interval widths, gate thresholds, blocking strengths, and tolerances settle to mutually consistent values that realize a valid cognitive function. On this reading emergence is the transition from locally formed, immature operators to a mature, composable circuit—a threshold effect in operator formation, like a phase transition that forms a crystal, not a mysterious statistical jump. We observe this directly

in our controlled training, where the emergence of a capability accompanies the formation of its operators and their circuit.

Scaling. Performance grows with cognitive-neural operator count and combination space: more parameters afford more recognition regions, more blocking paths, finer operator structure, more abstract high-level operators, and a larger space of operator compositions, so higher-dimensional concept regions and longer multi-step circuits become representable. Scaling regularities are then the macroscopic shadow of this growth in operator resources and its link to cognitive function. We observe this relationship between scaling and operators, and the boundary of functional extension, in our training experiments.

Generalization. A novel input that falls inside an existing condition interval opens the corresponding gate and fires an existing operator; variable slots let the bound entities change. This is exactly the behavior measured in Section 10: held-out valid combinations and unseen identity pairs succeed because the shared comparison and gate fire on new inputs that satisfy their conditions, not because those cases were memorized.

Hallucination. A reliable answer should require several sufficient conditions to hold together—an AND of gates. When those conditions are not fully formed, the gate degrades toward an OR, and a single mis-triggered condition can open the output path. Concretely, hallucination corresponds to parameter mismatch: a recognition interval too wide (an unknown input falls into a known region), a threshold too low, blocking too weak, an answer-gate bias too low, a tolerance too large, or an over-strong concretization that emits a fluent but unsupported answer. The remedies are correspondingly constructive and testable—tighten recognition intervals, raise answer-gate thresholds, strengthen negative-weight blocking, move from single-condition to AND-of-conditions gating, and add an explicit "insufficient evidence" path.

Multimodal alignment and in-context learning. Each modality maps through its own recognition into a shared concept space, after which transfer and gating are modality-independent; alignment is recognition into common concept values. In-context learning binds already-formed pattern-matching and transfer operators to the current variables rather than creating new operators: given a prompt of the form $[A][B]\ldots[A] \to ?$, the model recognizes the latent $A \to B$ pattern and calls the existing transfer operator on the new argument.

12. Relation to Interpretability Research: The Operators Behind Features and Circuits

The operator account is not a competitor to feature- and circuit-level interpretability but a structural layer above it: where that work *locates* features, subcomponents, and circuits, the operator account says what *functional role* each plays and why it must be present. Several independent findings correspond to the predicted operators.

Interpretable, causally effective monosemantic features [21] behave as recognition operators or their outputs: a feature activates when input and context fall into a characteristic

interval, mapping many surface forms to one concept value. Attribution graphs and sparse feature circuits [1, 15, 17] are the empirical information flow among staged operators—the reasoning circuits the account predicts must exist. Multilingual shared circuits [15] match "concept generation + shared transformation + concretization": surface forms in different languages map to a shared concept, run through one transformation, and are concretized per language. Multi-step reasoning, entity-recognition, and hallucination paths [15] match staged operator triggering and mis-gating, and induction heads [18] match latent-pattern recognition followed by calling an existing operator.

Read this way, the framework upgrades interpretability from locating structure to assigning it a mechanism—from "what does this feature represent?" to "which recognition operator, gate, comparison, or transfer path does it implement, and how do they compose?" It turns observed features and circuits into a debuggable map of operators. The correspondences are support for the account.

13. Significance and Applications: From Mechanism to Interpretability, Safety, and Industry

13.1. Scientific and theoretical significance

At the scientific level the account lowers reasoning from "symbolic rules" to "neuron computation": natural-language and formal-logic reasoning are shown to be realizable by neuron parameters and conditional gating structure, which makes reasoning, language, and other semantic transformation a matter of computable, interpretable, simulable, reproducible circuits rather than philosophy alone. It thereby reframes the long dispute over whether AI neural networks merely predict statistics or genuinely reason—statistics and logic are not opposed, because statistical training can form operators that encode logical regularity. The account also offers a unified foundation for connectionist and symbolic views, and, because the operators are defined by computation rather than substrate, a shared interpretive level for artificial and biological neural systems.

13.2. Interpretability and safety

As an interpretability tool, the framework upgrades analysis from reading attention maps to disassembling operator circuits, locating reasoning paths, hallucination sources, and erroneous matches, and turning a black box into a map that can be debugged, verified, and repaired. It suggests a principled route to hallucination control—tighten recognition and start intervals, raise answer-gate thresholds, strengthen negative-weight blocking, and move from single-condition to multiple parallel sufficient-condition gating. For AI safety and alignment it points to neural-operator-level tooling that may, as the work develops, finely trace large-model reasoning chains from concrete neuron parameters, cut dangerous paths, and implant safety-gating operators.

13.3. From analysis to design, and industrial outlook

The aim is not to revive hand-built rule-based AI but to expose the internal mechanism of neural cognition, thereby guiding the design of new neural-network models on the basis of that mechanism. As AI methods now decompose and predict protein structure, AI agents could, following the principles and construction here, decompose trained-model parameters into operators and conversely design new systems—potentially realizing reasoning by injecting neural reasoning operators directly. Deep learning has succeeded remarkably yet still lacks a mechanistic theory; just as thermodynamics advanced the steam and internal-combustion engine, a principled account of cognitive neural mechanism may advance technology and application. We report, in preliminary work to be detailed separately, small systems built on these principles that, in performing the same cognitive function, substantially reduce compute, show a plausible path to faster training, and achieve faster inference under theory-guided design. More importantly, the operator theory moves model architecture design from empirical search toward mechanistic calculation: the required cognitive operators and reasoning chains can be used to theoretically calculate model width, neuron count, and network depth, and to locate immature operators, layers, and local structures during training for targeted optimization. In training small models we find that the operator theory can guide the model's width and depth, greatly speeding up training compared with blind search, and that operator-level analysis can reveal where mature operators have not yet formed, allowing targeted local training that completes faster and improves performance. The prospective industrial value follows, stated as directions rather than established results: a compliance foundation for trusted AI in finance, law, medicine, and autonomous driving, where explainability, traceability, and verifiability are required; lower compute, inference time and training cost through partial operator design—a "neural editing" analogous to gene editing—that reduces parameters and power while strengthening reasoning; a core tool for AI safety and alignment through built-in safety neural operators; industrialization of vertical-domain models (legal reasoning, medical diagnosis, mathematical proof) by building in dedicated interpretable operators and clearly explaining the model's parameter-level reasoning and decision behavior; and a longer-range outlook toward novel chips, brain–computer interfaces, and human neural-repair engineering.

14. Conclusion

Cognitive neural operators are the mechanism by which neural networks realize semantic formation and semantic transformation in language and reasoning. Recognition, classification, concept generation, comparison, conditional gating, suppression, semantic transfer, output concretization, and so on are not merely descriptive labels; they can be realized by explicit weights, biases, ReLU-equivalent activations, and neural topology. The worked realization shows how these operators combine to transform a premise structure into a conclusion structure, and exact simulation confirms it. Controlled experiments further show that trained networks exhibit the theory-predicted operator structure, or realize the same operator functions as distributed, split, cross-shared, compressed, or stretched topological variants. This provides the first explicit and clearly interpretable per-neuron parameter-level realization of natural-language reasoning, and so opens the black box of emergent reasoning at the

level of mechanism rather than behavior. From the same mechanism the account explains large-model emergence, scaling, generalization, and hallucination, aligns with independent interpretability findings rather than competing with them, and points toward operator-level interpretability, hallucination control, AI-safety tooling, and the design of reasoning into networks. The theory has already shown guiding value for model design and training: in small-model practice, it helps set network width, neuron count, and depth in advance, locate under-trained layers and local structures by operator analysis, and then apply targeted optimization to improve training efficiency and overall performance. In these small models, theory-guided design also yields markedly faster reasoning and substantially lower compute. Because the operators are abstract, the same account gives a common mechanistic language for comparing symbolic-rule implementations with artificial and biological neural systems—reasoning lowered from symbolic rules to neuron computation, and the complexity of real models reread as a small set of operator laws unfolding in high-dimensional, distributed form.

References

[1] E. Amesien, J. Lindsey, A. Pearce, W. Gurnee, N. L. Turner, B. Chen, C. Citro, et al. Circuit tracing: Revealing computational graphs in language models. Anthropic Research, 2025. Mar. 27.

[2] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y.-T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[3] Ishita Dasgupta, Andrew K. Lampinen, Stephanie C. Y. Chan, Hannah R. Sheahan, Antonia Creswell, Dharshan Kumaran, James L. McClelland, and Felix Hill. Language models show human-like content effects on reasoning tasks, 2022.

[4] K. Dobs, J. Martinez, A. J. Kell, and N. Kanwisher. Brain-like functional specialization emerges spontaneously in deep neural networks. *Science Advances*, 8(11):eabn8931, 2022.

[5] Changde Du, Kaicheng Fu, Bincheng Wen, Yi Sun, Jie Peng, Wei Wei, Ying Gao, Shengpei Wang, Chuncheng Zhang, Jinpeng Li, Shuang Qiu, Le Chang, and Huiguang He. Human-like object concept representations emerge naturally in multimodal large language models, 2024.

[6] Jianyu Duan. A preliminary study on the principles of thinking in meaning activity. *Journal of Artificial Intelligence (in Chinese)*, 1987.

[7] Jianyu Duan. The current status and development of artificial intelligence. In *Proceedings of the National Conference on Thinking Science, China (in Chinese)*, China, 1988.

[8] Jianyu Duan. On the structural system of thinking science. In *Proceedings of the National Conference on Thinking Science, China (in Chinese)*, China, 1988.

[9] Jianyu Duan. Principles of thinking. In *Proceedings of the National Conference on Thinking Science, China (in Chinese)*, China, 1988.

[10] Jianyu Duan. Several issues on the study of thinking science. In *Proceedings of the National Conference on Thinking Science, China (in Chinese)*, China, 1988.

[11] A. Goldstein, H. Wang, L. Niekerken, M. Schain, Z. Zada, B. Aubrey, T. Sheffer, S. A. Nastase, H. Gazula, A. Singh, A. Rao, G. Choe, C. Kim, W. Doyle, D. Friedman, S. Devore, P. Dugan, A. Hassidim, M. Brenner, Y. Matias, O. Devinsky, A. Flinker, and U. Hasson. A unified acoustic-to-speech-to-language embedding space captures the neural basis of natural language processing in everyday conversations. *Nature Human Behaviour*, 9:1041–1055, 2025.

[12] J. Hu, M. A. Lepori, and M. Franke. Signatures of human-like processing in transformer forward passes, 2025.

[13] Aaron T. Kuan, Giulio Bondanelli, Laura N. Driscoll, Julie Han, Minsu Kim, David G. C. Hildebrand, Brett J. Graham, Daniel E. Wilson, Logan A. Thomas, Stefano Panzeri, Christopher D. Harvey, and Wei-Chung Allen Lee. Synaptic wiring motifs in posterior parietal cortex support decision-making. *Nature*, 627:367–373, Feb 2024.

[14] Yibei Li, Ethan J. Michaud, David D. Baek, Jack Engels, Xiangkun Sun, and Max Tegmark. The geometry of concepts: Sparse autoencoder feature structure, 2025.

[15] J. Lindsey, W. Gurnee, E. Amesien, B. Chen, A. Pearce, N. L. Turner, C. Citro, et al. On the biology of a large language model. Anthropic Research, 2025. Mar. 27.

[16] H. Luo and L. Specia. From understanding to utilization: A survey on explainability for large language models, 2024.

[17] Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models, 2025.

[18] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads, 2022.

[19] T. Räuker, A. Ho, S. Casper, and D. Hadfield-Menell. Toward transparent ai: A survey on interpreting the inner structures of deep neural networks, 2023.

[20] Hassan Sajjad, Nadir Durrani, and Fahim Dalvi. Neuron-level interpretation of deep nlp models: A survey, 2022.

[21] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, et al. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. Transformer Circuits Thread, Anthropic, 2024.

[22] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38, Feb 2024.