
REASONING AS STRUCTURE-PRESERVING TRANSFORMATION: A STRUCTURALIST FRAMEWORK FOR AGI

Qiyang Shen*

gmail@shu.edu.cn, 82113113a@gmail.com

December 11, 2025

ABSTRACT

We think that reasoning is most naturally modeled as structure-preserving transformations between representations, carried out under the invariant quantities that must be preserved for the reasoning to remain valid. In contrast to current large language models, we think a more efficient reasoning system requires an internal "structure space" whose elements are objects, admissible morphisms, and task-specific invariants. We refer to this perspective as Structure-Space Reasoning (SSR).

Within this space, a reasoning step corresponds to applying an admissible morphism that preserves the invariants characterizing the problem. We think that using such a structural representation space can substantially reduce the effective search space of reasoning while abstracting common reasoning pathways across different domains, enabling forms of compositionality and analogy that are difficult to realize with existing neural architectures.

Building on this view, we outline an AGI architecture in which: (1) large language models are used to extract structural patterns-along with their associated invariants from a wide range of reasoning processes; (2) a structure-representation layer encodes objects, morphisms, and invariants; (3) Adopting an AlphaZero-style search procedure, the structure search module selects transformations that preserve the problem's invariant quantities or produce new structures consistent with those quantities; (4) symbolic and numerical computation is delegated to external "compute modules", such as large scale integral computations, in order to approximate how mathematicians use computational tools to assist their reasoning.

1 Introduction

The progress of science and technology depends not only on experiments and engineering, but also on rigorous logical reasoning, which often guides the design of new experiments rather than merely interpreting their outcomes. In artificial intelligence, this has brought increasing attention to the capacity of a model to perform mathematical and logical reasoning. Recent mathematically oriented systems are typically built upon large language model (LLM) architectures and further refined through reinforcement learning, benefiting from vast working memory and high-quality, high-capacity parametric long-term memory enabling strong performance on many benchmark mathematical tasks. Nevertheless, we believe that if AI systems were equipped with a representational substrate intrinsically suited for reasoning—rather than relying on linguistic prediction alone—their capabilities could advance to a fundamentally higher level.

Multimodal models represent another active line of research. A widely held view is that, given sufficient computational resources, endowing an AI system with both a "language cortex" and a "visual cortex"—in loose analogy to human cognition—can substantially enhance its overall intelligence.

In this work, drawing on structuralism, category theory, and related developments in the foundations of mathematics and logic, we argue that achieving a more efficient reasoning model requires a third form of cortical substrate: a structure cortex. This analogy is purely functional, not biological. At the theoretical level, such a substrate should enable a

*Thanks to ChatGPT for providing unique perspectives during the writing of this article, and to the many great logicians, mathematicians, and engineers whose ideas have deeply influenced this work.

qualitative leap in reasoning capability; at the engineering level, it would allow far richer forms of inference within fixed computational budgets.

We think that current deep learning systems lack an explicit structure space, and therefore their ability to perform systematic reasoning remains fundamentally limited.

A very natural question is: what is the minimal representational substrate required for reasoning? Our earlier investigations suggested that systems restricted to purely logical tokens—connectives, quantifiers, and a small set of functional symbols—operate within an extremely compact embedding space that can isolate logical relations more cleanly than full natural language, but this "logic-only" representation proved fundamentally insufficient: although it reduces semantic noise, it cannot capture the diverse relational, geometric, algebraic, and diagrammatic structures that real reasoning routinely involves. Token-level logical representations excel at local inference but do not scale well to global structural abstraction, unless equipped with higher-level structure modules.

This observation led us to a broader conclusion: the difficulty lies not in the choice of tokens but in the representational space itself. Natural language produces an embedding space that is unnecessarily large and noisy, whereas a purely logical vocabulary collapses the space too aggressively, erasing essential structural distinctions. What is missing is a representational layer whose basic units are themselves *structures*, equipped with invariants and admissible transformations. This line of reasoning motivates the central hypothesis of the present work: The appropriate substrate for reasoning is neither linguistic nor exclusively logical; it is intrinsically structural.

Our central claim is that the essence of reasoning lies neither in symbols nor in language, but in structure-preserving transformations that maintain appropriate invariants.

With this perspective in mind, we propose the following overall strategy. A large language model (LLM) is first used to extract a broad collection of objects, morphisms, structural transformations, and invariants, which together serve as supervised training data for learning a structure-based representation. These components are organized into a unified structure space (O, M, I) .

On top of this representation, we introduce an AlphaZero-style mechanism for selecting the next reasoning step. The policy network chooses either (1) a structure-preserving transformation under a selected invariant, which serves as the next inference move, or (2) the construction of a new structure to be added to the structure space. The value network then evaluates whether the resulting new structure constitutes genuine progress toward solving the target problem, while Monte Carlo Tree Search (MCTS) explores sequences of structural transformations in a manner analogous to strategic game play.

Finally, we incorporate an external calculator module for performing explicit symbolic or numerical computations, and for returning the result in a single consolidated output.

Isolating heavy computation in this way allows the reasoning engine to focus exclusively on structural transformations and invariant-preserving logic, rather than on raw algebraic or analytic manipulation.

If reasoning can be carried out inside an explicit structure space, then inference becomes a form of computation rather than a linguistic act. Such a shift would make advanced reasoning—especially in mathematics—orders of magnitude more powerful, enabling discoveries far beyond the limits of human symbolic thought.

Contributions This paper makes three contributions: (1) a structuralist formulation of reasoning as invariant-preserving transformation; (2) an architectural proposal for AGI based on a structure space (O, M, I) ; (3) explicit training objectives (invariance, MDL, structure equivalence, next-structure prediction) that operationalize the framework.

2 Motivation

We now outline the motivation for introducing a structure space.

2.1 Category Theory and Structuralism

Throughout the history of mathematics, category theory and structuralism have played central roles. Their central insight is that the internal nature of an object is often less important than the relations it participates in; by focusing on these relations, one can abstract the structural patterns that govern reasoning. Grothendieck is perhaps the most celebrated exemplar of such "structural thinking": the strength of his approach lies in showing that seemingly different surface phenomena can be unified under a single underlying structure, allowing one to avoid re-deriving similar arguments in each individual case.

2.2 Modeling How Humans Reason About Mathematical Problems

When we examine how humans reason about mathematical problems, we observe that some concepts are naturally suited to linguistic or symbolic manipulation, such as equations or inequalities, while others are handled more intuitively through geometric or diagrammatic reasoning. For example, the notion of an induced topology becomes far clearer when one imagines a simple Venn diagram. In many cases, forming a mental image that "corresponds" to a textual description makes reasoning substantially easier than relying on text alone.

This difference arises because the visual system operates in a representational space that is high-dimensional, continuous and parallel, allowing it to encode geometric and causal relations directly. Language, by contrast, is linear, low-dimensional, discrete, and heavily compressed. Linguistic tokens require a sequence of mental steps—decompression, structural reconstruction, and manipulation. A sentence such as "a table in a room, a cup on the table, and a pen to the left of the cup" forces both humans and language models to incrementally construct a spatial model, whereas a visual system can obtain the underlying structural relations almost immediately.

But is the visual space itself the optimal substrate for reasoning? Not necessarily. Symbolic reasoning is often far more effective for algebraic or logical tasks, while purely visual representations cannot capture higher-dimensional or highly abstract structures.

In what space, then, can reasoning be made more efficient?

Our answer is that reasoning operates most naturally in a different kind of representational domain—a structure space.

In fact, visual space is not fundamentally a reasoning substrate, it merely provides one concrete instance of a structured representational domain. Our structure space generalizes this idea beyond the limitations of spatial geometry.

It is important to note that structure is not merely a correspondence between modalities (for example, between a sentence and an image). Over the course of a nontrivial reasoning process, the mapping between linguistic tokens, diagrams, and formal symbols may shift dynamically, but the underlying structure—comprising objects, morphisms, and invariants, remains the stable and essential element of the reasoning process.

In this sense, structure is inherently *cross-modal* and *dynamic*. It cannot be defined within any single modality; rather, it exists only within a separate structure space in which objects, morphisms, and invariants appear as first-class entities.

In our framework, language and vision provide modality-specific, and often noisy, projections of an underlying structural space. Genuine reasoning, however, operates not on these projections but on transformations within the structure space (O, M, I) .

Once a sufficiently rich structure space has formed, internal reasoning no longer needs to depend on human-readable labels; the relevant structural forms are the invariant patterns themselves. Moreover, as category theory suggests, reasoning processes across seemingly different domains may instantiate the same underlying structures. Extracting these latent structural patterns can greatly improve reasoning efficiency and allow structural knowledge learned in one domain to be transferred or shared between others.

Our proposed structure space (O, M, I) provides the foundational substrate for structure-space reasoning (SSR).

2.3 Limitations of LLM Embedding Space for Reasoning

A natural question is why enlarging the embedding space of a large language model often reduces its reasoning efficiency?

Intuitively, a human speaker faced with too many possible word choices may struggle to select the most appropriate expression. Language models behave similarly. Although reinforcement learning can push a mathematical specialist model to downweight irrelevant tokens (such as metaphysical, emotional, or conversational terms), these tokens still occupy positions in the embedding matrix and therefore expand the search space during inference. As a result, large embedding spaces expand the hypothesis class of token predictions, which indirectly burdens reasoning processes that rely on token-level decisions.

Does this imply that shrinking the embedding matrix would yield better reasoning performance? Clearly not.

Reasoning is not purely linguistic. If every geometric or algebraic concept must be represented solely through first-order logical primitives, the representational burden becomes prohibitive. For example, defining a circle entirely in logical or set-theoretic terms requires:

- defining the real numbers,
- defining point sets,

- defining coordinate systems,
- defining distance,
- defining the circle equation.

Under such a representation, any inference involving circles must manipulate extremely verbose logical expansions, making the reasoning process inefficient and brittle. The failure of both large and small embedding spaces suggests that the problem is not the size of the token vocabulary but the ontological type of the representational unit. A structure space replaces tokens with structured entities, avoiding both semantic noise and representational collapse. What is needed instead is a way to capture the *structural essence* of geometric objects. A circle, for instance, can be defined structurally as

"the set of all points at equal distance from a given center."

This definition highlights the invariant that characterizes a circle. Yet such structural abstractions cannot be encoded directly in a language model’s embedding matrix, because the matrix contains only a fixed vocabulary of discrete tokens. It cannot store "a new definition," "a relational structure," or "a set of invariants."

From a reasoning standpoint, however, these abstract structures *should* exist as selectable primitives. They are precisely the kinds of entities that belong in a structure space rather than a linguistic token space.

Although large language models can update the meaning of token vectors, they are still not efficient enough.

Thus, the limitations of LLMs stem not merely from vocabulary size but from a deeper representational mismatch: the embedding matrix is optimized for natural language, not for structural abstraction. A more capable reasoning system must represent structures as first-class entities and manipulate them via admissible morphisms that preserve their invariants. From this perspective, existing reasoning paradigms each capture only part of the picture. Large language models operate primarily as pattern approximators over a largely linear embedding space. They can absorb many statistical regularities, but they lack explicit access to the invariants that support reliable long-horizon, compositional, and cross-domain reasoning. Graph neural networks, by contrast, encode local relational structure very naturally, yet struggle to construct higher-order "structures of structures" that organize multiple levels of abstraction. At the other extreme, formal logical systems can manipulate explicit structures with full rigor, but have no built-in mechanism for autonomously inventing new structures or discovering higher-level invariants. A structure space aims to integrate the strengths of these approaches while addressing their weaknesses, by making structures and invariants first-class entities that can be learned, transformed, and evolved.

2.4 The Strong Analogy Between AlphaZero and Human Mathematical Reasoning

Once reasoning steps are formalized as morphisms, they naturally form a discrete action space. This turns reasoning into a sequential decision process, to which AlphaZero-style planning is a natural fit.

When we examine how humans carry out mathematical reasoning, we find that it draws on both computational tools (e.g., calculators or symbolic engines) and genuine logical inference. The latter, in its pure form, is strikingly reminiscent of AlphaZero: one may view the current set of available structures as analogous to a board position in a strategic game. A reasoning step then corresponds to choosing a structural transformation or constructing a new structure, much like selecting a move.

However, large language models are not efficient engines for such exploration. Because their internal search is entangled with the constraints of linguistic prediction, they often fail to navigate directly toward the correct chain of inference. This motivates the use of an explicit scoring mechanism: each action—whether introducing a new structure or applying an invariant preserving transformation—updates the state of the structure space, and a learned value function evaluates the resulting state. In this way, structural reasoning can proceed via an AlphaZero-style policy-and-value framework rather than relying solely on token level prediction.

2.5 Relation to Existing Paradigms

Our proposal is related to several existing lines of work but differs in focus. Neuro-symbolic systems and automated theorem provers manipulate explicit symbols or formal proofs, but typically lack an explicit, learnable structure space in which objects, morphisms, and invariants are represented as first-class entities and can be autonomously extended. Graph neural networks capture local relational structure, yet they usually operate on a fixed graph and are not equipped with an explicit notion of task-specific invariants or theorem-like high-level morphisms. Large language models with tool use and external calculators provide powerful interfaces for calling symbolic or numerical engines, but their internal representations remain token-centric and do not separate structural reasoning from linguistic prediction.

Structure-Space Reasoning (SSR) differs from these approaches by treating *structures and invariants themselves* as the basic objects of computation. The structure space $\mathcal{X} = (O, M, I)$ is both a representational layer and a reinforcement learning environment: it can grow new objects, morphisms, and invariants over time, while a policy-and-value system learns to navigate this space via high-level, invariant-preserving morphisms.

3 The Important Roles of Invariance and Structure in the Reasoning Process

3.1 Invariance in the Reasoning Process

Reasoning does not operate on "content" itself; rather, it moves between structures while preserving the invariants that define the problem. Across a wide range of reasoning scenarios, one observes a common pattern: reasoning always involves change. Figures are modified in diagrammatic reasoning, and symbolic expressions are rewritten in formal logic. Yet certain aspects must remain invariant. At its core, reasoning consists of transformations that are admissible within a given structural context, and these transformations must preserve invariants that maintain the identity of the problem under consideration. Invariance is what keeps reasoning valid; controlled change is what allows reasoning to progress.

Most reasoning steps preserve some invariants, while possibly modifying others. What matters is that the task-relevant invariants remain preserved. For example, to derive B from A , the conclusion B must remain within the same underlying structure determined by A ; otherwise the chain of reasoning becomes ill-formed. This requirement of "preservation" is precisely what invariants encode. If a model cannot detect which aspects of a representation must remain fixed under transformation, it cannot learn what makes two steps of reasoning belong to the same problem. Invariants serve as the fulcrum of reasoning—they are what allow a logical chain to stay coherent.

We may formalize this idea as follows. Let a "state" S denote any form of cognitive representation: mathematical formulas, natural language expressions, diagrams, graphs, geometric shapes, programs. Reasoning is then a constrained transformation within this representational space. A reasoning process can be viewed as a sequence

$$S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow \dots \quad (1)$$

where each step satisfies

$$I(S_k) = I(S_{k+1}). \quad (2)$$

These invariants ensure that although the surface form of a representation may change, the core structural content is preserved. Thus reasoning should be understood not as surface-level symbol manipulation, but as structure-preserving transformation.

3.2 Structure Space

Reasoning occurs in structure space, not in language space.

Genuine reasoning does not unfold within linguistic space, but within what we call a structure space. In many cases, human visual reasoning often feels more intuitive because the visual system already provides a low-dimensional representational domain endowed with geometric properties such as continuity, adjacency, and spatial invariance. These properties make many forms of geometric or topological reasoning naturally accessible when carried out visually.

However, visual reasoning has intrinsic limitations: it struggles to represent higher-dimensional relations or highly abstract structures. For such tasks, humans naturally shift to symbolic or hybrid forms of representation. The usefulness of vision does not arise because it is the fundamental substrate of reasoning, but because it happens to embody *one particular* structural regime—namely, geometric structure.

The aim of constructing a structure space is to generalize this advantage: to separate what can change from what must remain invariant, and to collect the essential elements of reasoning into a unified representational domain.

By "structure space" we mean a space that contains structural objects, the admissible transformations between them, and the invariants preserved by those transformations.

Reasoning in such a space requires both local transformations between individual objects and global transformations between entire structures. Therefore, the structure space must support morphisms at multiple levels.

Not every map is a valid inference rule; only transformations that preserve selected invariants—and that correspond to admissible steps within a reasoning process—qualify as reasoning.

Formally, we define a structure space as a tuple

$$\mathcal{X} = (O, M, I), \quad (3)$$

where O is the set of objects (structures), M is the set of morphisms (allowed transformations), and I is the set of invariants that these morphisms preserve.

A Simple Example. To make the notion of invariance in reasoning more concrete, consider the elementary setting of parity. Let the representational state encode the parity of an integer. Suppose the initial state is

$$S_1 : n \text{ is even}$$

The allowed morphisms are the transformations

$$M = \{ n \mapsto n + 2, n \mapsto n - 2 \},$$

that is, operations that shift an integer by two units. The invariant preserved by these morphisms is simply

$$\text{parity}(n) \in \{\text{even}, \text{odd}\}$$

which serves as the invariant throughout the transformation

$$I(S_k). \quad (4)$$

This gives rise to a reasoning sequence

$$S_1 \rightarrow S_2 \rightarrow S_3,$$

$$S_1 : n \text{ is even} \longrightarrow S_2 : (n + 2) \text{ is even} \longrightarrow S_3 : (n + 4) \text{ is even},$$

where each state S_k corresponds to applying one of the allowed morphisms. Although the integer itself changes, its parity is preserved at every step, ensuring that the invariant condition

$$I(S_k) = I(S_{k+1})$$

holds throughout the sequence.

The reasoning process here can therefore be understood as a sequence of structure-preserving transformations within a structure space

$$X = (O, M, I)$$

where O is the set of integers equipped with parity labels, M is the set of allowable parity-preserving transformations, and I is the parity invariant. In practice, each object in O , M , I , X is encoded into a representation that can be accessed by both neural networks and search algorithms, ensuring that structural transformations, scoring, and planning can operate on the same underlying representation.

This simple example illustrates how even the most elementary forms of reasoning rely on transforming representations while maintaining structural constraints.

Below we provide further examples of invariance and structure in reasoning:

Structure Type	Typical Invariant	Why Reasoning Holds
Logical Structure	Transitivity of implication	Ensures consistent conclusions
Symmetry Structure	Equivalence under group actions	Preserves identity under transformations
Algebraic Structure	Preservation of algebraic laws	Guarantees valid operations
Topological Structure	Invariance under continuous deformation	Maintains the "same shape"
Order Structure	Preservation of ordering relations	Supports causal/temporal inference
Graph Structure	Preserves node–edge relations	Enables compositionality and dependency tracking
Geometric Structure	Distances/angles remain invariant	Supports geometric reasoning
Conceptual Structure	Stability of semantic relations	Basis for human abstract reasoning

Table 1: Structural Types, Typical Invariants, and Why Reasoning Holds

3.3 A Layered Model of Formal and Semantic Structures

A natural question concerns the status of the "structures" that inhabit the space $\mathcal{X} = (O, M, I)$. Must they be fully specified mathematical objects with rigorously verifiable properties, or may the system also operate with semantic, approximate, or partially justified structures? We think that a realistic reasoning architecture must incorporate both, but in a layered fashion.

At the lowest layer, the system operates on objects whose structural properties are mathematically rigorous and whose transformations admit precise verification. These include algebraic structures, graph-theoretic objects, logical formulas, and proof fragments whose invariants can be checked exactly. This layer functions analogously to a "formal core" of reasoning: it enforces logical soundness, guarantees invariant preservation, and prevents invalid morphisms.

Above this formal layer, however, it is necessary to allow more flexible, semantic structures that are not fully reducible to strict mathematical verification. Human reasoning routinely employs such structures: for example, assuming the generalized Riemann Hypothesis enables a large body of meaningful conclusions in analytic number theory, even though the hypothesis itself is unproven. Similarly, mathematicians often reason with diagrams, analogies, or heuristic patterns that capture genuine structural regularities long before they are formalized. In our framework, higher layers of the structure space may contain semantic structures distilled from corpora of reasoning—such as recurrent proof patterns, heuristic variables, implicit analogies, and high-level argument shapes. These structures may not always be verifiable in the strict logical sense, but they remain useful as long as: (i) they admit admissible transformations M , (ii) they maintain task-relevant invariants I in an approximate or statistical sense, and (iii) they yield effective guidance for exploration, compression, or analogy.

This layered design reflects the way human mathematical practice integrates rigor with heuristic insight. The formal layer ensures correctness, whereas the semantic layers perform heuristic and intuitive forms of reasoning—generating conjectures, posing tentative hypotheses, and exploiting loose structural analogies. Crucially, each structure—whether formal or semantic—is encoded into a representation accessible to both neural networks and search, allowing the system to integrate strict logic with heuristic pattern discovery within a unified framework.

4 A Roadmap Toward AGI: Architecture Based on Structure-Space

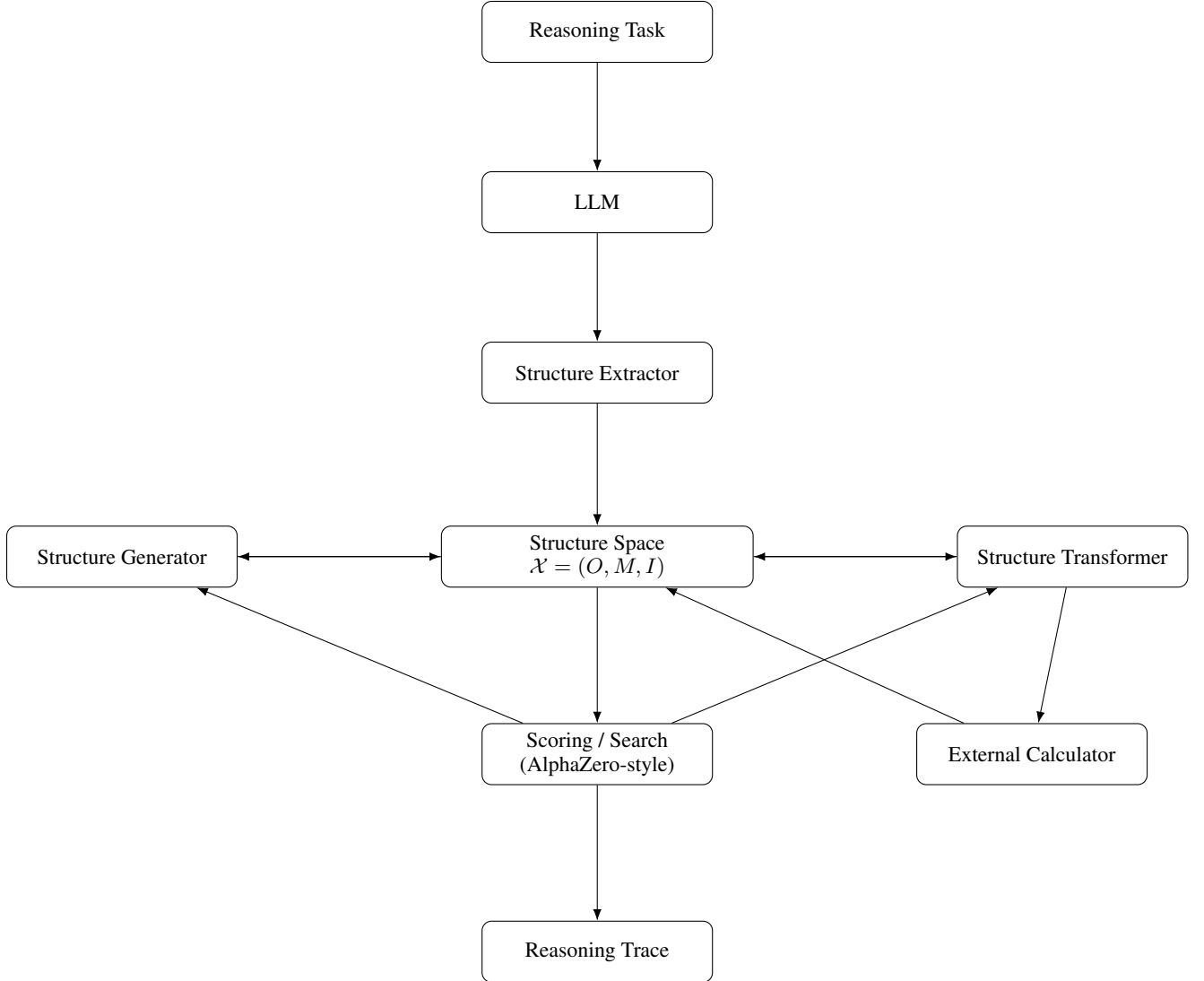


Figure 1: Overview of the proposed structure-based reasoning architecture.

4.1 Overview of Structure-Space Reasoning (SSR)

We refer to this overall framework as **Structure-Space Reasoning (SSR)**: a computational paradigm in which reasoning is performed not in token space but within an explicit structure space equipped with objects, morphisms, and invariants. Operationally, SSR treats a reasoning trajectory as a sequence of invariant-preserving morphisms in a structure space:

$$X_0 \xrightarrow{m_1} X_1 \xrightarrow{m_2} \dots \xrightarrow{m_T} X_T,$$

where each $X_t \subseteq (O, M, I)$ is a structural state and every admissible morphism m_t satisfies $I(X_t) = I(X_{t+1})$ for the task-relevant invariants I .

4.2 The Role of Structure Space in Reasoning

A general artificial intelligence system grounded in a structure space forms the central component of our proposed framework. Unlike traditional symbolic systems, the structure space is open-ended and can expand dynamically as

the system discovers new structures, morphisms, and invariants. Such a system functions as a reasoning-specialized module whose role is not to store content, but to operate on invariants and structure-preserving transformations. Under this view, diverse kinds of mathematical, logical, or geometric structures can be unified within a single representational scheme—consisting of objects, morphisms, and invariants—which in turn supports a mechanism for structural transformation, structural comparison, and invariant-based learning.

4.3 Reasoning as Structure-Preserving Search

Once reasoning steps are modeled as morphisms, the process becomes a sequential decision problem. Furthermore, the close parallel between logical reasoning and the game-like search performed by AlphaZero suggests a natural computational strategy. One may regard the current configuration as the "state" of a game and each admissible morphism as a possible move. This analogy motivates carrying out reasoning in an AlphaZero-like fashion: reinforcement-learning-based search over a space of structure-preserving transformations. We expect that algorithms inspired by AlphaZero can be adapted to this setting, potentially enabling automatic discovery of mathematical reasoning steps.

4.4 Modular Computation and Automatic Module Evolution

In practice, a structure-based reasoning system is unlikely to take the form of a single monolithic network. Human mathematical cognition relies on several interacting subsystems: (i) a "calculator" module capable of exact symbolic or numerical computation (such as evaluating integrals or solving differential equations), (ii) a reasoning module that supports forward, backward, and "jumping" inferences over abstract structures, and (iii) an intuitive modeling module that constructs geometric or physical analogies.

An important example comes from the relationship between the number of solutions of an elliptic curve over a finite field and the size of that field. A key parameter governing this relationship is constrained by the *Hasse–Weil bound*:

$$|\#E(\mathbb{F}_p) - (p + 1)| \leq 2\sqrt{p}.$$

Such a constraint was originally observed through plotting and extensive computation: the geometric shape becomes apparent long before one encounters a formal proof. Yet this visually accessible phenomenon inspired deep mathematical investigation, ultimately contributing to the chain of ideas that led to the proof of Fermat’s Last Theorem.

Motivated by this cognitive picture, our framework assumes a family of interacting modules rather than a single end-to-end Transformer. Importantly, the internal parameter spaces of these modules need not resemble conventional token-embedding vector spaces. Instead, each internal state may be regarded as a richly structured "reasoning step"—one that encodes, for example, whether an external computation should be invoked, the inputs and outputs of such calls, its local reasoning context, and its nesting relations with other steps.

4.5 Core Modules of the SSR Architecture

We envision the system as comprising several key modules:

Structure Representation and Structure Space: a module that encodes structures directly. All structures live inside an extensible structure space that stores objects, morphisms, invariants, and their interrelations.

Structure Generator: proposes latent structures underlying a given problem and expands the search space with new candidates.

Structure Transformer: applies structure-preserving transformations that maintain selected invariants. This module functions as the core reasoning engine.

Scoring Module: evaluates all existing structures—including objects, morphisms, invariants, and reasoning chains—by assigning learned scores. This scoring mechanism is implemented via neural networks. The scoring module takes as input a structural configuration and outputs scalar evaluations representing progress, complexity reduction, or invariant satisfaction. To avoid unbounded expansion, structures may be pruned using MDL-based scoring and invariant consistency checks.

External Calculator: invoked when reasoning requires heavy numerical computation, providing fast results that would otherwise require long step-by-step derivations.

LLM: a large language model used to extract implicit structures, morphisms, objects, and invariants from large corpora of mathematical or logical reasoning traces. LLMs serve as powerful priors for structure induction, yet the system must impose explicit structural constraints and validation procedures to guarantee that the extracted reasoning patterns are rigorous, invariant-preserving, and truly relevant to the underlying problem.

In this picture, the scoring module subsumes the role of a structure validator: it decides which candidate structures satisfy the required invariants and deserve to be retained. The structure space itself functions as a structure library, storing reusable objects, morphisms, and invariant patterns, analogous to a theorem library but at a more abstract level.

Together, these components define the computational substrate on which structural reasoning is carried out.

4.6 Neural Representation of the Structure Space

A central practical challenge in realizing a structure-space AI is determining how the elements of the structure space—objects, morphisms, and invariants—are to be encoded in a learnable, extensible neural substrate. Neural architecture choice and representational format directly determine whether the system can faithfully capture structural relations and support invariant-preserving transformations.

We argue that graph neural networks (GNNs) provide the most natural starting point for implementing the structure space. GNNs are inherently suited to representing objects and relations, and they offer a flexible mechanism for encoding invariants directly on nodes, edges, or higher-order relational configurations. In contrast to conventional fixed-topology neural networks, a structure-space system must allow its representational capacity to expand as new structures, morphisms, and invariants are discovered. This suggests that the underlying neural architecture should support forms of *self-expanding* computation, enabling the model to grow new nodes, edges, or modules as the structure space evolves. Dynamically extensible architectures are essential for any system built atop an open-ended, continually expanding structure space.

Concretely, each structural configuration can be represented as a graph in which nodes correspond to objects (or substructures), edges correspond to morphisms, and their attributes encode the relevant invariants. A single element of O may itself be represented as a subgraph, while elements of M are typed edges or higher-order links connecting such subgraphs; I is represented as constraints or learned features attached to these components. The GNN operates over this graph, allowing both local message passing and global aggregation to capture how objects, morphisms, and invariants interact.

Within this design, GNNs function not as the final architecture but as an initial substrate capable of capturing the combinatorial anatomy of (O, M, I) , while self-expanding neural mechanisms enable the system to reconfigure and augment its internal structure in response to the emergence of new objects, morphisms, and invariants. This establishes the representational foundation on which the subsequent reasoning modules operate.

4.7 Potential Parameter Sharing Between LLMs and Structure-Space Encoders

Large language models and graph-based structure encoders are trained under different objectives, yet both ultimately learn representations that organize information into latent structure spaces. Although their internal mechanisms differ, there may exist partially overlapping information—such as locality, compositionality, or attention-based weighting that may allow certain parameters or encoding patterns to be reused or exchanged between the two systems.

At present, it is not known which components of an LLM’s representation are suitable for reuse inside a structure-space encoder. However, it is reasonable to treat the LLM as providing an initial parameterization or feature prior: some of its embedding vectors, attention weights, or intermediate activations may supply useful starting points for the GNN-based encoder. The structure encoder would then refine these parameters under explicit structural constraints and invariant-preserving objectives.

Whether deeper forms of parameter sharing are possible—such as aligning attention mechanisms or identifying shared subspaces—remains an open research question. For now, we regard LLM-derived features as optional priors that may accelerate learning, while the core structural representations are learned directly within the structure space.

4.8 From Token Sequences to Structural Representations

In practice, the system must convert textual reasoning traces into candidates for (O, M, I) . Given a corpus of proofs, solutions, and derivations, an LLM first segments each trace into units such as definitions, lemmas, theorem statements, and individual inference steps. For each segment, the LLM identifies: (i) the objects involved (nodes in the eventual graph), (ii) the transformations or logical links (candidate morphisms), (iii) the constraints that should remain fixed across steps (candidate invariants).

These extracted elements are then canonicalized into graph-structured representations: objects are converted into nodes with type and attribute annotations; morphisms become labeled edges or higher-order links; and invariants are encoded either as global constraints or as local attributes on the relevant subgraphs. The resulting graphs serve as the raw input to

both the GNN-based structure encoder and the scoring module, which determines which candidates should be admitted into the structure space \mathcal{X} .

4.9 Initialization of the Structure Space

A natural question is how the structure space should be initialized. Since the system cannot begin with an empty universe of objects and morphisms, we require an initial library of powerful structures that can serve as high-impact reasoning primitives. Large language models provide a practical mechanism for extracting such structures: by analyzing large corpora of mathematical and scientific reasoning, an LLM can identify definitions, theorems, constructions, and transformation patterns that exhibit unusually high generality and cross-domain reuse.

Which structures should be considered "maximally powerful"? In our framework, these are the structures that (i) compress large amounts of reasoning into compact transformations, (ii) reappear across diverse domains, and (iii) function as high-level "jump operators" that enable substantial shortcuts in reasoning trajectories. The Fundamental Theorem of Calculus (Newton–Leibniz theorem) is a canonical example. When viewed as a structure, this theorem encapsulates an extremely high-level morphism: it collapses long sequences of limit manipulations into the simple relation

$$\int_a^b f(x) dx = F(b) - F(a),$$

thereby unifying differential and integral structures under a single invariant. Although a purely logical description of this structure would be highly complicated, once it is placed on top of existing structures such as area, derivatives, and accumulation, it becomes an exceptionally powerful reasoning tool—indeed one of the strongest elements of the initial structure space. More generally, the initial structure space should contain those abstractions that offer the largest structural "leverage": compact, reusable, invariant-preserving patterns that support reasoning across broad conceptual territories.

4.9.1 Constructing the Initial Structure Space

The structure space cannot begin as a blank slate. To support nontrivial reasoning, the system requires an initial library of structures, namely those that exhibit strong generality, cross-domain applicability, and high MDL compression power. We outline a principled pipeline for constructing such an initial structure space.

Step 1: Extracting Candidate Structures via LLM-Based Mining. A large language model is used to analyze corpora of mathematical proofs, textbooks, problem solutions, and scientific derivations. The LLM identifies: (i) frequently reused patterns, (ii) canonical transformations, (iii) definitions with high cross-domain applicability, (iv) theorems that collapse long reasoning chains, (v) structural analogies across algebraic, geometric, logical, and physical domains.

Each extracted candidate is represented not by its textual form but by a structural signature: the objects involved, the relations imposed, and the invariants preserved.

Step 2: Scoring and Filtering Candidates. Each candidate structure is evaluated with respect to:

1. **MDL compression power:** how much it shortens common reasoning chains.
2. **Invariant strength:** whether it specifies robust, widely useful invariants.
3. **Cross-domain reuse:** appearance frequency across mathematical or scientific domains.
4. **Combinatorial leverage:** whether it enables the construction of new morphisms.

Only structures above a threshold score are retained.

Step 3: Canonicalization and Graph Encoding. Each selected structure is encoded as a graph with:

- nodes representing objects or substructures,
- edges representing morphisms,
- annotated invariants representing constraints that must be preserved.

They constitute the basic underlying representation units of SSR. Strong theorems and high-level identities can also enter the structure space as powerful morphisms that collapse long reasoning chains into compact invariant relations.

Step 4: Bootstrapping and Closure. Finally, the initial structure space is closed under: (i) composition of morphisms, (ii) admissible transformations, (iii) structural equivalence operations (e.g., isomorphisms), (iv) automatically generated substructures. This produces a coherent, extensible universe of objects, morphisms, and invariants from which SSR can begin operating.

Together, these steps construct an initial structure space that is compact yet extremely high in reasoning leverage, enabling the system to conduct nontrivial structural reasoning from the outset.

4.10 Learning Objectives and Reinforcement Learning in Structure Space

Given that these modules correspond to distinct types of structural operations, it is natural to ask how they can be learned rather than hand-designed.

4.10.1 Training Objectives in Structure Space

To identify which structures are truly meaningful, the system must rely on principled scoring mechanisms. Reinforcement learning over the structure space can further refine these mechanisms. Here "meaningful" refers to structures that preserve task-relevant invariants and reduce reasoning complexity. Broadly, the system should be encouraged to preserve invariants under allowed transformations, to favor shorter reasoning chains when possible, and to learn the next reasoning state—whether a new structure or a further inference step—through supervised signals.

More concretely, we introduce explicit training objectives defined over the structure space and combine them with reinforcement learning. These objectives incentivize the model to maintain relevant invariants, compress reasoning trajectories, and discover structurally equivalent representations of the same underlying problem.

Concretely, we may introduce the following families of training objectives:

Loss 1: Invariant Consistency Given an input and a set of allowed transformations, the internal structural representation should remain unchanged under those transformations. This encourages the system to filter out nuisance variation and focus on the invariant core of the problem.

Loss 2: Minimal Description Length (MDL) When multiple structural explanations account for the same reasoning chain, the system is rewarded for choosing compositional structures with minimal description length. This biases the model toward reusable primitives and compact abstractions. In mathematical practice, many profound theorems serve exactly this function: they compress long sequences of elementary steps into a single structural leap. The Newton–Leibniz theorem (the fundamental theorem of calculus) is a canonical example—transforming repeated limit manipulations into a direct correspondence between differentiation and integration. In the structure space, any structure (together with its associated reasoning chains) that enables such large leaps should receive higher reward, as it yields a dramatic MDL reduction. In our framework, such theorems correspond to high-level morphisms that collapse long chains of low-level transformations.

Loss 3: Next-Structure Prediction (Self-Supervised) Given the current structural state, the model predicts the next state in a reasoning trajectory—for example, the next node in a proof tree, the next morphism composition, or the next transformation in a causal or algebraic graph. This objective trains a dynamic structural model over the tuple (O, M, I) . Crucially, the prediction target is not a token sequence but a structural state within (O, M, I) .

Loss 4: Structure-Equivalence Learning (Contrastive) Structures that instantiate the same invariant pattern—regardless of their surface form, modality, or domain—should lie close in representation space, while structurally unrelated configurations should be pushed apart.

These losses are complementary rather than mutually exclusive; in practice, combinations of them may be used.

4.10.2 Reinforcement Learning over the Structure Space

Once reasoning is formulated as a search over structure-preserving morphisms, the structure space (O, M, I) naturally induces a reinforcement learning environment. Each structural state encodes a set of objects, morphisms, invariants, and partial reasoning trajectories, while each action corresponds to introducing a new structure, applying an admissible morphism, composing existing morphisms, or querying an external calculator.

State. A state s_t consists of: (i) a current structure configuration $X_t \subseteq (O, M, I)$, (ii) a partial reasoning chain represented as a composition of morphisms, (iii) auxiliary metadata such as invariant satisfaction, MDL scores, and structural equivalence classes. The state is encoded using a GNN-based representation that captures relational dependencies between objects, morphisms, and invariants.

Action Space. Actions correspond to valid structural operations, including:

1. Applying a morphism $m \in M$ to an object or structure.
2. Composing morphisms $m_1 \circ m_2$ when admissible.
3. Select or generate candidate structures from the structure generator.
4. Applying a high-level theorem-like morphism (e.g., the Newton-Leibniz transformation).
5. Calling the External Calculator to supply a missing computation.

Each action must preserve the invariants relevant to the current reasoning task.

Policy Network. A policy network $\pi_\theta(a \mid s)$ proposes the next morphism or structural operation. Its input is the GNN-encoded state, and its output is a probability distribution over admissible transformations.

Value Network. A value network $V_\phi(s)$ estimates long-horizon rewards, including: (i) progress toward solving the target problem, (ii) reduction in reasoning-chain length (MDL), (iii) satisfaction of task-relevant invariants, (iv) discovery of reusable structures useful across tasks.

Rollouts and MCTS. Monte-Carlo Tree Search explores reasoning trajectories by repeatedly simulating:

$$s_{t+1} = f(s_t, a_t), \quad a_t \sim \pi_\theta(a \mid s_t).$$

The tree search balances exploitation of high-scoring morphisms with exploration of novel structural transformations, exactly as in AlphaZero but operating on the combinatorial structure space instead of a game board.

Reward. A reward function aggregates structural objectives:

$$R(s_t, a_t) = \lambda_1 \cdot \text{InvariantConsistency} + \lambda_2 \cdot \text{MDLReduction} + \lambda_3 \cdot \text{ProblemProgress} + \lambda_4 \cdot \text{StructureReuse}.$$

This encourages the discovery of high-level structural shortcuts, lemma-like patterns, or transformations that collapse large reasoning chains.

Overall, reinforcement learning over the structure space enables the system to autonomously construct reasoning strategies, compress theories, and discover powerful morphisms without explicit supervision.

This completes a unified roadmap in which reasoning emerges from searching, transforming, and evaluating structures under the invariants that define the problem.

5 Discussion and Limitations

The framework outlined in this paper remains at a high level. Many key components are still not detailed, and require further work. First, we have not fixed a concrete algorithm for extracting and canonicalizing structures from raw corpora; different choices of abstraction granularity may lead to very different structure spaces. The trade-off between strict formal verification and heuristic semantic structures is only sketched at the architectural level. Precisely characterizing this trade-off—and designing principled interfaces between a formally verified core and approximate higher layers—remains an open challenge. Second, the computational complexity of search in an open-ended structure space is potentially large; scalable approximations, and task-specific constraints will likely be required. Finally, the concrete design of SSR models—and their integration with large language models—still presents many open challenges.

References

- [1] S. Mac Lane. *Categories for the Working Mathematician* (2nd ed.). Springer, 1998.
- [2] S. Shapiro. *Philosophy of Mathematics: Structure and Ontology*. Oxford University Press, 1997.
- [3] J. C. Baez and M. Stay. Physics, topology, logic and computation: A rosetta stone. In B. Coecke (ed.), *New Structures for Physics*, pages 95–172. Springer, 2011.
- [4] P. W. Battaglia, J. B. Hamrick, V. Bapst, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- [5] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [6] J. B. Tenenbaum, C. Kemp, T. L. Griffiths, and N. D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011.
- [7] J. Pearl. *Causality: Models, Reasoning, and Inference* (2nd ed.). Cambridge University Press, 2009.
- [8] D. Silver, J. Schrittwieser, K. Simonyan, et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815*, 2017.
- [9] D. Silver, T. Hubert, J. Schrittwieser, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.
- [10] DeepSeek-AI Team. DeepSeekMath: Enhancing Mathematical Reasoning in Large Language Models via Reinforcement Learning. *arXiv preprint arXiv:2402.03300*, 2024.