MIXAL: A Neural-Symbolic Architecture for Computational Conceptualization and ARC Reasoning

Tanvi Pooranmal Meena

July 10, 2025

Abstract

We present *MIXAL* (Memory-augmented, Iterative, eXplainable, Abstraction Learner), a novel neural-symbolic architecture that addresses the symbol grounding problem through computational conceptualization. The system combines learned transformation primitives with memory-augmented multi-step reasoning to achieve human-aligned few-shot generalization on visual reasoning tasks, particularly the Abstraction and Reasoning Corpus (*ARC*). Our approach combines offline learning of visual-to-symbolic pattern mappings from public *ARC* data with online hypothesis generation, validation, and refinement guided by a relevance-filtered memory control system. *MIXAL* addresses two fundamental limitations of current *ARC* systems: the inability to systematically discover and compose transformation primitives, and context loss during multi-step reasoning chains. Through iterative rule refinement and explicit state management, *MIXAL* achieves human-aligned few-shot generalization while maintaining full interpretability of its reasoning process. The core innovation lies in mathematicizing human conceptualization through structured representations that bridge raw perception and symbolic reasoning.

Keywords: Few-shot learning, Neural-symbolic reasoning, Abstraction and reasoning, Memory-augmented systems, Interpretable AI, Symbol grounding, Computational conceptualization

1 Introduction

The Abstraction and Reasoning Corpus (*ARC*) presents a fundamental challenge in artificial intelligence: achieving human-level few-shot generalization on novel visual reasoning tasks. Unlike traditional machine learning benchmarks that benefit from large-scale training data, *ARC* tasks provide only 1-4 input-output examples and require systems to infer and apply transformation rules to novel inputs without prior exposure to the specific task type.

Current approaches to *ARC* reasoning fall into two categories: monolithic neural models that struggle with systematic generalization, and hand-crafted rule systems that lack the flexibility to discover novel patterns. Both approaches often fail to capture the human-like ability to rapidly form, test, and refine hypotheses about underlying transformation rules.

More fundamentally, they lack a principled approach to the symbol grounding problem - the challenge of connecting raw perceptual data to meaningful symbolic representations.

Key Breakthrough Understanding: Computational Conceptualization

We observe that current AI systems lack a conceptual layer between raw perception and symbolic reasoning. This creates a fundamental gap:

- **Low-level**: Raw sensory data (pixels, tokens)
- **High-level**: Ungrounded symbolic patterns
- **Missing**: Mathematical conceptual representations

Humans don't do pattern matching - they do rule derivation through visual analysis

We observe that humans solve *ARC* tasks not through mere pattern matching, but by actively deriving underlying rules through visual analysis. This process involves visual parsing (identifying shapes, colors, relative positioning), input-output mapping ("Where is the input in the output?"), difference analysis ("What changed and how?"), rule hypothesis ("All red/yellow squares get 4 corner squares"), validation by applying the hypothesized rule to subsequent examples, and refinement based on success or failure. This human-centric approach, emphasizing rule derivation and iterative refinement, motivates our system design.

Since humans can solve these tasks rapidly through visual inspection, the underlying transformation space must be finite, learnable, and systematically composable. This insight motivates our approach: learn the space of transformation primitives from available data, then dynamically compose and refine these primitives on a per-task basis through structured conceptual representations.

1.1 Core Innovation: Computational Conceptualization

MIXAL addresses the symbol grounding problem by mathematicizing human conceptualization through formalizing three fundamental perceptual primitives:

- **Shape**: Algebraic signatures, topological invariants, graph structures
- **Color**: Symbolic tokens with semantic transformation maps
- **Relative Positioning**: Affine transformations, relational graphs, spatial algebras

Our contributions are fourfold:

1. A systematic methodology for mining transformation families from *ARC* data and mapping visual patterns to symbolic rules.
2. A Conceptual Representation Interface (*CRI*) that bridges raw perception and symbolic reasoning through structured mathematical representations.
3. A memory-augmented reasoning architecture that maintains coherent context across multi-step transformation chains through relevance-filtered retention.
4. An iterative refinement framework that improves initial rule hypotheses through systematic validation and adaptation.

2 Related Work

2.1 ARC Reasoning Systems

Early *ARC* approaches relied heavily on hand-crafted domain-specific languages (DSLs) for expressing transformation rules. While these systems achieved reasonable performance on tasks matching their programmed primitives, they struggled with novel combinations and subtle variations.
Recent neural approaches have applied large language models and vision transformers to *ARC* tasks, achieving improved flexibility at the cost of interpretability. However, these systems often fail on tasks requiring precise geometric reasoning or multi-step logical chains.

2.2 Neural-Symbolic Integration

The integration of neural perception with symbolic reasoning has shown promise across various domains. Our approach builds on this tradition by using neural networks for pattern recognition and symbolic systems for rule execution and composition, bridging them through structured conceptual representations.

2.3 Memory-Augmented Neural Networks

Previous work on memory-augmented systems has focused primarily on sequence modeling and meta-learning. We extend these concepts to multi-step reasoning by introducing relevance-filtered memory mechanisms that enable systematic exploration of reasoning paths while maintaining cognitive realism.

2.4 Symbol Grounding

The symbol grounding problem - connecting symbolic representations to perceptual experience - has been a long-standing challenge in AI. Our approach provides a systematic solution through computational conceptualization, formalizing perceptual primitives as manipulable mathematical structures.

3 Problem Formulation

An *ARC* task T consists of a set of training examples $E = \{(I_1, O_1), (I_2, O_2), \ldots, (I_n, O_n)\}$ where each I_i is an input grid and O_i is the corresponding output grid, plus a test input I_{test}. The goal is to predict O_{test} such that the transformation rule applied to generate each (I_i, O_i) pair also maps I_{test} to O_{test}.
The key challenges are:

- **Few-shot generalization**: Learning from minimal examples (typically 1-4).
- **Novel task types**: No exposure to the specific transformation during training.
- **Compositional reasoning**: Many tasks require multi-step transformation chains.
- **Precise execution**: Visual reasoning requires exact grid-level accuracy.

- **Symbol grounding**: Connecting visual patterns to meaningful symbolic representations.

4 MIXAL Architecture

4.1 System Overview

MIXAL operates in two phases: offline learning of transformation primitives and online per-task reasoning. The architecture follows the pipeline:
Perception → *CRI* → *RRL* → Reasoning
(continuous) → (structured concepts) → (symbolic operations) → (composable logic)
The offline phase mines the public *ARC* dataset to discover recurring transformation families and trains a Rule Proposer (*RP*) to map visual patterns to symbolic rules. The online phase applies this learned knowledge to solve novel tasks through iterative hypothesis refinement, mediated by structured conceptual representations.

4.2 Conceptual Representation Interface (CRI)

The *CRI* serves as the crucial bridge between raw perception and symbolic reasoning. It provides structured mathematical representations of visual scenes that are both learnable by neural networks and interpretable by symbolic systems.

4.2.1 Structure

The *CRI* represents visual scenes as structured objects with mathematical properties:

```
{
  "objects": [
    {
      "id": "obj_1",
      "shape": "rectangle",
      "color": "red",
      "position": {"x": 2, "y": 3},
      "size": [3, 1],
      "relations": ["above(obj_2)", "aligned(left, grid_edge)"]
    }
  ]
}
```

4.2.2 Mathematical Foundations

The *CRI* mathematicizes perceptual primitives through:

Modality	Representation
Shape	Graphs, algebraic signatures, topologies
Color	Symbolic tokens + semantic transformation maps
Position	Affine transformations, relational graphs

This structured approach enables systematic manipulation of visual concepts through symbolic operations while maintaining grounding in perceptual reality.

4.3 Offline Learning Components

4.3.1 Pattern Mining Module (*PMM*)

The *PMM* analyzes the public *ARC* dataset to identify recurring transformation motifs. Rather than learning task-specific solutions, *PMM* discovers reusable transformation families that appear across multiple tasks.

The mining process focuses on atomic transformations to ensure reliable extraction:

- **Geometric transformations**: rotation, reflection, translation, scaling.
- **Color operations**: mapping, highlighting, pattern filling.
- **Spatial relationships**: alignment, grouping, connection.
- **Pattern completion**: symmetry, repetition, extrapolation.

Each discovered family is parameterized to capture the range of variations observed in the training data. For example, rotation operations are parameterized by angle, center point, and object selection criteria. Complex compositions emerge through the iterative refinement process rather than explicit offline learning.

Key Components of Visual Analysis:

To facilitate the human-inspired reasoning process, the *PMM* and subsequent online components rely on the following key visual analysis capabilities:

1. **Visual Feature Extractor**: This component performs low-level perception, including:
   - Object segmentation: Identifying connected components within the grid.
   - Shape classification: Recognizing common shapes (e.g., rectangle, L-shape).
   - Color inventory and mapping: Cataloging colors present and their transformations.
   - Spatial relationships: Determining how objects are positioned relative to each other (e.g., inside, adjacent, corner-of).
2. **Input-Output Mapper**: This is a crucial insight borrowed from human reasoning. It aims to "Find input pieces in output" by:
   - Identifying preserved vs. transformed elements.
   - Detecting additions, deletions, and modifications between the input and output grids.
3. **Transformation Analyzer**: This component interprets the differences identified by the mapper into actionable changes, such as:
   - Color changes: e.g., red $\rightarrow$ blue, or "preserve non-red."
   - Spatial changes: e.g., "+4 corners," "rotate 90°," "mirror horizontally."
   - Pattern changes: e.g., "fill interior," "extend edges."
4. **Rule Synthesizer**: This final component converts these detailed visual differences into symbolic rules. It involves:
   - Parameterizing transformations: e.g., add_corners(color=X, count=4).
   - Generating conditional logic: e.g., if color in [red, yellow] then add_corners.

4.3.2 Rule Proposer Training (*RPT*)

The Rule Proposer is a neural encoder-decoder model trained to map visual pattern changes to symbolic transformation rules. The training data consists of (input_grid, output_grid, transformation_rule) tuples derived from the pattern mining process.
The encoder processes visual differences between input and output grids through the *CRI*, while the decoder generates symbolic expressions using the discovered transformation families. This design ensures that proposed rules are both grounded in visual patterns and expressible in the learned symbolic vocabulary.

4.4 Online Reasoning Components

4.4.1 Rule Proposer (*RP*)

Given the first training example (I_1, O_1), the *RP* generates a ranked list of candidate transformation rules. The *RP* leverages its training on public *ARC* data to map visual patterns to symbolic expressions from the learned transformation families.
The proposal process begins with simple single-step transformations and relies on the refinement loop to discover necessary compositions. Rules are ranked by confidence scores that reflect both visual pattern matching and prior probability from the training distribution.

4.4.2 Rule Executor (*RE*)

The *RE* applies symbolic transformation rules to conceptual representations through the *CRI*. Unlike end-to-end neural approaches, symbolic execution provides precise, interpretable transformations with predictable behavior.
The executor supports atomic operations and dynamically discovered composite operations through the refinement process. Error handling mechanisms detect invalid operations and provide detailed feedback for rule refinement, including pixel-level accuracy, structural similarity, and semantic consistency measures.

4.4.3 Rule Refiner (*RR*)

When initial rule hypotheses fail validation, the *RR* generates improved variants through systematic modification strategies:

- **Parameter adjustment**: Tuning continuous parameters within learned bounds.
- **Rule composition**: Combining multiple transformation families sequentially.
- **Abstraction level changes**: Moving between specific and general rule formulations.
- **Error-guided search**: Using validation feedback to direct refinement.

The *RR* maintains systematic tracking of refinement attempts and validation results, enabling iterative improvement through structured exploration of the hypothesis space.

4.4.4 Memory Control System (*MCS*)

Unlike traditional memory systems, the *MCS* implements cognitively realistic selective retention through relevance-filtered memory:

$$\text{score}(H) = \alpha \cdot \text{validation\_success} + \beta \cdot \text{generality} + \gamma \cdot \text{reuse} - \delta \cdot \text{error\_penalty}$$

Key Features:

- Dynamic pruning of unhelpful reasoning paths
- Task-sensitive retrieval of useful hypotheses
- Generalization compression promoting recurring patterns into templates
- Version control for systematic hypothesis exploration

4.4.5 MIXAL Reasoning Loop

The *MIXAL* inference process follows an iterative hypothesis-test-refine loop:

1. **Parse I/O examples → CRIs**: Convert visual grids to structured conceptual representations.
2. **Propose symbolic transformation rules via RP**: Generate candidate rules from the first example.
3. **Execute rules on input CRI → predicted output**: Apply rules through the *CRI* interface.
4. **Validate against actual output**: Compare predicted and actual outputs.
5. **Refine rule via RR if validation fails**: Generate improved variants using error feedback.
6. **Apply validated rule to test input**: Execute final rule on test case.

This approach reduces system complexity while maintaining the core neural-symbolic integration and iterative refinement capabilities that distinguish *MIXAL* from existing approaches.

5 Rule Representation Language (RRL) Design

The core of *MIXAL*'s interpretability and generalizability lies in its expressive yet precisely defined Rule Representation Language (*RRL*). This symbolic vocabulary serves as the bridge between the system's visual understanding and its executable actions, reflecting the human process of rule derivation through visual analysis.

5.1 Core Design Principles

Our *RRL* is designed with the following principles to ensure it is both powerful and practical for *ARC* tasks:

- **Compositional**: Complex transformations are constructed by combining simpler, atomic primitives.
- **Parameterized**: Operations can be applied with varying parameters (e.g., angle of rotation, specific colors).
- **Conditional**: Rules can specify conditions under which operations apply, based on object properties.
- **Interpretable**: The language is designed to be human-readable, allowing for clear understanding and debugging of the inferred rules.

- **Executable**: Rules have a direct and deterministic mapping to grid manipulation operations within the *RE*.

5.2 Hierarchical Vocabulary Structure

The *RRL* is structured hierarchically to facilitate progressive abstraction and systematic rule generation, mirroring the multi-faceted nature of visual reasoning:

5.2.1 Level 1: Object Operations

These are fundamental transformations applied directly to identified objects or pixel sets within the grid:

- `rotate(object, angle)`: Rotates an object by a specified angle (e.g.,`rotate(red_square, 90)`).
- `reflect(object, axis)`: Mirrors an object across a given axis (e.g., `reflect(blue_L, horizontal)`).
- `translate(object, dx, dy)`: Shifts an object by specified horizontal and vertical displacements (e.g., `translate(yellow_dot, 2, -1)`).
- `scale(object, factor)`: Resizes an object by a given factor (e.g., `scale(green_rect, 2)`).
- `recolor(object, new_color)`: Changes the color of an object or specified pixels (e.g., `recolor(red_pixels, blue)`).

5.2.2 Level 2: Spatial Operations

These operations concern the positioning, alignment, and arrangement of objects relative to the grid or other elements:

- `align(object, reference, edge)`: Positions an object relative to a reference (e.g., `align(square, grid, top_left)`).
- `place_at(object, position)`: Places an object at a specific coordinate or named position.
- `surround(object, pattern)`: Encloses an object with a specified pattern or border (e.g., `surround(red_square, blue_border)`).
- `fill_inside(object, color)`: Fills the interior region of a shape (e.g., `fill_inside(rectangle, yellow)`).
- `extend(object, direction, length)`: Extends a linear object in a given direction (e.g., `extend(line, right, 3)`).

5.2.3 Level 3: Pattern Operations

These primitives capture more complex visual regularities and completions:

- `replicate(object, pattern)`: Repeats an object according to a specified pattern (e.g., `replicate(square, 3x3_grid)`).
- `symmetrize(object, axis)`: Creates a symmetrical counterpart of an object (e.g., `symmetrize(shape, vertical)`).

- `complete_pattern(partial)`: Infers and completes a larger pattern from a partial input
  (e.g., `complete_pattern(half_circle)`).
- `connect(obj1, obj2, style)`: Draws a connection between two objects
  (e.g., `connect(dots, straight_line)`).

5.2.4 Level 4: Conditional Logic

These operations enable rules to apply selectively based on object properties or grid states:

- `select(condition)`: Filters objects based on specified criteria (e.g., `select(color==red)`).
- `for_each(objects, operation)`: Applies an operation to every object satisfying a condition
  (e.g., `for_each(red_squares, add_corners)`).
- `if_then(condition, operation)`: Executes an operation only if a given condition is met
  (e.g., `if_then(size>2, recolor(blue))`).
- `preserve(condition)`: Explicitly retains elements that satisfy a condition
  (e.g., `preserve(color!=red)`).

5.2.5 Level 5: Compound Operations

These allow chaining of operations:

- `sequence([op1, op2, op3])`: Executes operations in sequence
  (e.g., `sequence([rotate(90), translate(1,0), recolor(blue)])`).
- `parallel([op1, op2])`: Executes operations in parallel
  (e.g., `parallel([recolor(red, blue), add_border])`).

5.3 Concrete Example

Consider the human reasoning insight: "All red and yellow squares get 4 corner squares." This can be directly translated into our *RRL* as:

```
for_each(
  select(color in [red, yellow] and shape == square),
  add_corners(color=black, count=4)
)
```

5.4 Rule Grammar (BNF)

```
Rule ::= ConditionalRule | SimpleRule | CompoundRule

ConditionalRule ::= ‘if_then(‘ Condition ’,‘ Operation ’)’
SimpleRule ::= Operation
CompoundRule ::= ‘sequence([‘ RuleList ’])’ | ‘parallel([‘ RuleList ’])’
RuleList ::= Rule (‘,’ Rule)*
```

```
Operation ::= Transform | Spatial | Pattern
Transform ::= ‘rotate(‘ Object ’,‘ Angle ’)’
           | ‘reflect(‘ Object ’,‘ Axis ’)’
           | ‘translate(‘ Object ’,‘ Offset ’)’
           | ‘scale(‘ Object ’,‘ Factor ’)’
           | ‘recolor(‘ Object ’,‘ Color ’)’

Spatial ::= ‘align(’ Object ’,‘ Reference ’,‘ Edge ’)’
         | ‘place_at(‘ Object ’,‘ Position ’)’
         | ‘surround(‘ Object ’,‘ Pattern ’)’
         | ‘fill_inside(‘ Object ’,‘ Color ’)’
         | ‘extend(’ Object ’,‘ Direction ’,‘ Length ’)’

Pattern ::= ‘replicate(‘ Object ’,‘ Pattern ’)’
         | ‘symmetrize(’ Object ’,‘ Axis ’)’
         | ‘complete_pattern(‘ Partial ’)’
         | ‘connect(‘ Object ’,‘ Object ’,‘ Style ’)’

Condition ::= PropertyFilter | SpatialFilter | CompositeFilter
PropertyFilter ::= Property Operator Value
SpatialFilter ::= SpatialProperty Operator Value
CompositeFilter ::= Condition (‘AND’ | ‘OR’) Condition | ‘NOT’ Condition
Object ::= ‘select(‘ Condition ’)’ | ObjectReference
```

5.5 Execution Engine Interface

```
class RuleExecutor:
    def execute(self, rule: Rule, input_grid: Grid) -> Grid:
        """Executes symbolic rule on input grid."""

    def validate(self, rule: Rule, input_grid: Grid, expected_output: Grid) -> bool:
        """Checks if applying rule produces expected output."""

    def explain(self, rule: Rule) -> str:
        """Returns human-readable explanation of rule."""
```

6 Mathematicizing Conceptual Abstraction

While MIXAL introduces a robust pipeline bridging perception and symbolic reasoning, a deeper formalization is necessary to enable machines to reason at the level of human conceptual abstraction. At present, even interpretable AI systems rely on manually defined symbolic vocabularies or neural heuristics without a principled mathematical model of abstract thought. We argue that to achieve true conceptual understanding, machines must

operate over mathematically defined structures that reflect the way humans represent, manipulate, and reason about abstract concepts.

6.1 The Problem: From Concept to Computation

Human cognition operates not directly on pixels or tokens, but on structured, abstract mental models—concepts such as “symmetry," “container," “group," or “force." These concepts are not discrete labels but flexible, compositional, and often modality-independent constructs grounded in embodied experience and iterative generalization. Current symbolic systems encode such ideas via ad hoc rules, and neural systems often reduce them to latent vectors with no compositional structure. To bridge this gap, we propose a computational framework for **mathematicizing conceptual abstraction**—formally modeling abstract concepts as manipulable mathematical objects suitable for symbolic execution.

6.2 Core Hypothesis: Conceptual Algebra as Cognitive Substrate

We posit that abstract human concepts can be represented as elements in structured mathematical spaces, with transformations and relationships defined as morphisms or operators within those spaces. We identify several promising formal substrates:

- **Conceptual Spaces** [**?**]: Concepts are modeled as convex regions in high-dimensional spaces, with dimensions corresponding to perceptual qualities (e.g., color, shape, size). Similarity becomes a geometric relation.
- **Category Theory**: Concepts are objects in a category; transformations and analogies are morphisms. Functorial mappings encode analogy and generalization across domains.
- **Topological Signatures**: Qualitative perceptual invariants (e.g., connectedness, holes, symmetry) are formalized using algebraic topology—capturing the persistent features of concepts under deformation.
- **Typed Lambda Calculus and Modal Logic**: Conditional, probabilistic, or context-sensitive concepts can be expressed via dependent types or modal logic, enabling finer-grained reasoning about conceptual applicability.

6.3 Toward a Conceptual Compiler

To integrate these mathematical representations into a reasoning system, we envision a *Conceptual Compiler* that translates high-level, human-aligned conceptual structures into executable symbolic rules. Such a compiler operates over:

1. **Concept Definitions**: Types and signatures defining the structure of a concept (e.g., `SymmetricShape`, `ColorGroup`).
2. **Concept Transformations**: Morphisms that map concepts to each other (e.g., `rotate`, `mirror`, `contract`).

3. **Concept Composition**: Rules for building higher-order abstractions via function composition, categorical products, or logical conjunction.

This compiler links MIXAL's Conceptual Representation Interface (CRI) with the Rule Representation Language (RRL), effectively grounding perceptual input into an algebra of manipulable cognitive primitives.

6.4 Implications for Artificial Conceptualization

By formalizing abstract human concepts into mathematical structures, we create a bridge between intuitive cognition and machine reasoning. This enables:

- **Compositional Generalization**: Machines can reason about unseen combinations of known concepts.
- **Human-AI Alignment**: Conceptual models become inspectable, editable, and teachable.
- **Cross-Domain Transfer**: Abstract concepts like symmetry, hierarchy, or causality become domain-agnostic.
- **Continuous Self-Improvement**: New concepts can be synthesized through algebraic composition and refinement.

This formalization provides a principled foundation for MIXAL and future systems targeting robust, interpretable, human-aligned intelligence. It represents not just an engineering solution but a paradigm shift in how machines may come to understand.

7 Key Innovations

7.1 Computational Conceptualization

MIXAL provides a systematic solution to the symbol grounding problem through a mathematically grounded framework that bridges perception and reasoning. This is achieved by embedding abstract conceptual knowledge into structured, manipulable forms suitable for symbolic execution. The key elements include:

- **Mathematical Conceptualization**: Perceptual primitives—shape, color, position—are formalized as elements in structured mathematical spaces such as topological graphs, vector spaces, and symbolic algebras. More abstract cognitive constructs (e.g., symmetry, enclosure, adjacency, repetition) are treated as higher-order morphisms, forming a *Conceptual Algebra* that reflects human-like reasoning.
- **Structured Bridging**: The *CRI* serves as a computational layer that translates low-level visual input into high-level conceptual types. These types are rigorously defined using type signatures, geometric invariants, and logical constraints, making them both learnable by neural modules and operable by symbolic systems.

- **Compositional Reasoning**: The *RRL* encodes these conceptual representations as composable, parameterized symbolic rules. The hierarchical vocabulary structure supports reasoning over atomic operations, spatial configurations, pattern transformations, and conditional logic, all grounded in the formal Conceptual Algebra.
- **Conceptual Compiler (Future Extension)**: A forthcoming extension involves a compiler that translates human-aligned conceptual transformations—expressed in mathematical or logical form—into executable symbolic rules within *RRL*. This bridges human abstract thought with machine reasoning, enabling teachable, transferable, and generalizable cognitive capabilities.
- **Cognitive Alignment**: The *MCS* implements relevance-filtered retention and hypothesis management, enabling the system to explore complex reasoning chains while retaining conceptual coherence. It mirrors selective human memory processes and supports incremental abstraction discovery.

7.2 Learned Transformation Families

Unlike hand-crafted DSLs, *MIXAL* discovers transformation primitives directly from data. This ensures that the symbolic vocabulary covers patterns that actually occur in *ARC* tasks while remaining compact and interpretable. Discovered families are parameterized to support compositional generalization and grounded in mathematically defined perceptual structures.

7.3 Visual-to-Symbolic Rule Mapping

The trained Rule Proposer bridges the gap between visual pattern recognition and symbolic rule generation through structured conceptual representations. This hybrid approach combines the flexibility of neural networks with the precision of symbolic systems, explicitly incorporating visual parsing, difference analysis, and rule synthesis. Each proposed rule is grounded in a shared Conceptual Algebra, ensuring semantic consistency across tasks.

7.4 Memory-Augmented Multi-Step Reasoning

The relevance-filtered memory system enables systematic exploration of complex reasoning chains without context loss while maintaining cognitive realism. This addresses a fundamental limitation of current neural approaches that struggle with compositional reasoning. The memory system not only stores transformation hypotheses but also supports versioning, reuse, and abstraction compression—turning recurring conceptual patterns into reusable symbolic templates.

7.5 Iterative Hypothesis Refinement

Rather than requiring perfect initial rule generation, *MIXAL* improves hypotheses through systematic validation and refinement. This mirrors human problem-solving strategies and increases robustness to imperfect initial proposals. Refinement is guided by error signals, structural feedback, and concept-space similarity metrics, allowing MIXAL to converge on high-fidelity rules with minimal examples.

8 Experimental Design

8.1 Training Data Strategy

MIXAL leverages the public *ARC* dataset (1000+ tasks) for offline learning while maintaining strict separation from private evaluation data. The pattern mining process extracts transformation families without memorizing task-specific solutions.
Data augmentation strategies include:

- **Systematic parameter variation**: Generating rule variants with different parameters.
- **Composition synthesis**: Creating complex transformations by chaining learned primitives.
- **Negative example generation**: Learning from failed transformation attempts.

8.2 Evaluation Methodology

Performance evaluation focuses on:

- **Accuracy**: Exact match on private test sets.
- **Interpretability**: Human understanding of generated rules.
- **Efficiency**: Computational resource usage.
- **Generalization**: Performance on task types not seen during training.

8.3 Ablation Studies

Planned ablations examine:

- **Component contributions**: Individual module impact on overall performance.
- **Memory system effectiveness**: Performance with and without *MCS*.
- **Conceptual representation impact**: Comparison with direct pixel-level processing.
- **Transformation family coverage**: Impact of different pattern mining strategies.
- **Refinement strategies**: Effectiveness of different rule modification approaches.

9 Expected Contributions

9.1 Technical Contributions

- **Novel neural-symbolic architecture** for few-shot visual reasoning that effectively combines neural pattern recognition with symbolic execution, all mediated by structured conceptual representations. This significantly advances the state-of-the-art in bridging these two paradigms for complex reasoning tasks.
- **Systematic and formalized solution** to the long-standing symbol grounding problem through the innovative concept of computational conceptualization, which mathematically defines perceptual primitives and enables their structured manipulation.
- **Memory-augmented reasoning framework** featuring a relevance-filtered Memory Control System

(*MCS*) that ensures coherent multi-step inference, minimizes context loss, and aligns with cognitive principles of selective attention and memory retention.
- **Robust and scalable methodology** for autonomously mining fundamental transformation primitives and their families directly from diverse visual reasoning data, ensuring the system's symbolic vocabulary is empirically grounded and adaptable.

9.2 Research Impact

- **Advancing Interpretable AI**: By maintaining a transparent and human-readable reasoning process, *MIXAL* directly addresses the "black box" problem in AI, making its few-shot learning decisions fully auditable and understandable.
- **Informing Cognitive Modeling**: Provides a concrete, computational framework for understanding and simulating human-like problem-solving strategies, particularly in the domain of visual abstraction and reasoning.
- **Deepening Symbol Grounding Research**: Offers a principled and systematic approach to connecting raw perception to abstract symbolic representations, pushing the boundaries of how AI systems can truly "understand" their environment.
- **Enabling Broad Transfer Learning**: The learned conceptual representations and transformation families are designed to be abstract and generalizable, holding significant potential for transfer to other visual reasoning tasks and even other domains beyond the *ARC* corpus.
- **Demonstrating Modular AI Excellence**: Serves as a compelling case study for the effective integration of diverse AI modules (perception, memory, symbolic reasoning, refinement) into a cohesive and powerful system.

9.3 Practical Applications

Beyond achieving competitive performance in *ARC* benchmarks, *MIXAL*'s unique architecture and capabilities could significantly benefit a range of real-world applications:

- **Educational Technology**: Developing intelligent tutoring systems that not only provide answers but also explain the underlying reasoning steps, helping students grasp abstract concepts and problem-solving methodologies.
- **Scientific Discovery**: Aiding researchers in hypothesis generation, pattern recognition, and the formulation of new theories in data-limited or visually complex scientific domains by identifying underlying rules from minimal observations.
- **Robotic Planning and Control**: Enabling robots to rapidly adapt to novel manipulation tasks and environments by inferring operational rules from few examples, leading to more flexible and autonomous systems.
- **Automated Program Synthesis**: Facilitating the learning and generation of programming patterns or code snippets from minimal input-output examples, potentially accelerating software development and reducing manual coding effort.
- **Intelligent Design and Creativity Tools**: Assisting designers and artists by understanding and applying abstract visual rules to generate new patterns, layouts, or compositions based on high-level conceptual inputs.

10 Limitations and Future Work

10.1 Current Limitations

- **Pattern Mining Completeness**: While robust, the current pattern mining process relies on the public *ARC* dataset, which may not encompass every conceivable transformation type. Novel and highly idiosyncratic transformations might still pose a challenge.
- **Linear Refinement Constraints**: The iterative refinement process primarily explores hypotheses sequentially. This linear exploration might occasionally miss optimal solutions that could be found more efficiently through parallel or more sophisticated search strategies across the rule space.
- **Initial Rule Proposer Quality**: Although mitigated by the powerful iterative refinement loop, the overall system performance remains somewhat dependent on the quality of the initial hypotheses generated by the Rule Proposer. Improving the diversity and accuracy of initial proposals is an ongoing area of research.
- **Composition Discovery**: While the system can chain simple operations through refinement, the discovery of entirely novel, complex multi-step transformations as single, named entities (i.e., abstracting sequences into new primitives) primarily emerges through the iterative refinement process rather than explicit offline learning of complex compositions.
- **Scalability to Larger Grids/Higher Dimensionality**: While *ARC* grids are relatively small, extending *MIXAL* to much larger visual inputs or higher-dimensional data (e.g., 3D object manipulation) would require further research into efficient conceptual representation and reasoning at scale.

10.2 Future Directions

- **Enhanced Memory Systems**: Further developing the *MCS* to include more sophisticated mechanisms for selective context retention, potentially incorporating hierarchical memory structures or more dynamic weighting of historical data based on task similarity and long-term utility.
- **Learning from Failed Attempts**: Systematically incorporating insights from failed refinement attempts and invalid rule proposals to guide future hypothesis generation and refine the Rule Proposer itself, moving beyond simple error-guided search to more intelligent negative learning.
- **Automated Primitive Discovery**: Exploring methods for the system to autonomously discover *new* fundamental transformation primitives or composite operations that are not explicitly present in the initial mined set, allowing for even greater adaptability to unforeseen task types.
- **Beyond Visual Reasoning**: Investigating the applicability of the computational conceptualization framework and the overall *MIXAL* architecture to other domains requiring few-shot generalization and interpretable reasoning, such as natural language understanding, logical puzzles, or even scientific hypothesis generation.
- **Interactive Human-AI Collaboration**: Developing interfaces that allow human

users to provide feedback, corrections, or hints during the reasoning process, making *MIXAL* a collaborative tool for problem-solving rather than a fully autonomous agent. This could also aid in generating richer training data through human demonstrations.
- **Probabilistic Rule Induction**: Integrating probabilistic graphical models or Bayesian inference techniques into the rule generation and refinement process to explicitly quantify uncertainty in rule hypotheses and propagate it through the reasoning chain, leading to more robust and explainable predictions.
- **Self-Correction and Self-Improvement**: Designing mechanisms where the system can analyze its own reasoning failures and adapt its internal components (e.g., the *RPT* or *RR*) to prevent similar errors in the future, fostering continuous self-improvement.

10.3 Addressing Key Architectural Challenges

We anticipate and proactively address several core challenges in building a scalable, generalizable *ARC* reasoning system, ensuring *MIXAL*'s robustness and efficiency:

10.3.1 1. Rule Proposer Robustness and Refinement

While the Rule Proposer (*RP*) generates initial hypotheses, *MIXAL* is designed to thrive even with imperfect initial proposals. We integrate a feedback-driven iterative refinement loop managed by the Rule Refiner (*RR*), which systematically improves initial hypotheses across the 1–4 training examples. This mechanism, guided by detailed validation feedback from the Rule Executor (*RE*) and supported by structured state management within the Memory Control System (*MCS*), allows even initially flawed rules to converge toward valid and precise transformations.
In essence, failed hypotheses are leveraged as critical data points for learning and correction, rather than being discarded.

10.3.2 2. Managing Compositional Discovery Complexity

The *MIXAL* architecture efficiently discovers rule compositions through iterative refinement, rather than relying solely on explicit offline learning of every possible composite rule. This approach reduces the burden on the offline learning phase while still enabling the construction of complex multi-step transformations:
- **Systematic Sequential Composition**: Complex transformations are built step-by-step through successive refinement cycles, leveraging the hierarchical structure of the Rule Representation Language (*RRL*).
- **Efficient Refinement Budget Management**: The Memory Control System (*MCS*) plays a crucial role in balancing the depth of hypothesis exploration with computational efficiency by dynamically pruning unpromising reasoning paths.
- **Error-Guided Composition Strategies**: Validation feedback provides precise guidance for combining and modifying rules, directing the search towards effective compositions that address identified discrepancies.

These mechanisms ensure tractable composition discovery while maintaining the iterative improvement paradigm essential for few-shot generalization.

10.3.3 3. Ensuring Implementation Scalability and Efficiency

The design of *MIXAL* inherently supports practical deployment and scalability, optimizing for computational efficiency while managing complex reasoning processes:

- **Structured Memory Management**: The *MCS* employs relevance-filtered memory, avoiding the pitfalls of unbounded memory growth and high-dimensional vector spaces typical of end-to-end neural models. It intelligently retains only useful hypotheses and reasoning paths.
- **Interpretable Rule Execution**: The symbolic Rule Executor (*RE*), operating on structured Conceptual Representation Interface (*CRI*) outputs, provides precise, deterministic transformations. This avoids the computational overhead and non-determinism often associated with purely neural execution.
- **Focused Iterative Refinement**: While exploration occurs, the refinement process is guided by explicit error signals and systematic modification strategies, concentrating computational resources on the most promising areas of the hypothesis space.

This design enables practical deployment while maintaining extensibility for future enhancements without compromising core functionality.

10.3.4 4. Extensibility and Future Capability Expansion

The *MIXAL* architecture is designed with clear pathways for systematic enhancement and expansion beyond its current capabilities:

Immediate Enhancement Pathways (already in scope):

- **Rule Refiner Adaptivity**: Further enhancing the Rule Refiner's composition strategies, parameter optimization, and search heuristics to accelerate convergence and discover more complex patterns.
- **Enhanced Pattern Mining**: Expanding the coverage and sophistication of the Pattern Mining Module (*PMM*) to discover an even broader array of transformation families and their parameterized variations.
- **Advanced CRI Features**: Exploring more sophisticated mathematical foundations and representations within the *CRI* to capture nuanced visual properties and relationships.

Longer-Term Research Directions (consistent with Future Work):

- **Probabilistic Rule Induction**: Incorporating explicit uncertainty and probabilistic reasoning into rule generation and validation to handle ambiguity and improve robustness.
- **Automated Primitive Discovery**: Researching methods for *MIXAL* to autonomously synthesize entirely new fundamental transformation primitives or abstract higher-level concepts beyond the initially mined set.
- **Cross-Domain Transfer**: Investigating the application of *MIXAL*'s computational conceptualization and reasoning framework to non-visual domains, showcasing its potential for broader artificial general intelligence challenges.
- **Human-in-the-Loop Interaction**: Developing interfaces that allow for intuitive human feedback and guidance during the reasoning and refinement process, fostering collaborative AI problem-solving.

This layered approach ensures the system remains functional and competitive at each development stage while enabling systematic capability expansion towards more advanced intelligent behaviors.

11 Conclusion

MIXAL represents a significant and novel approach to few-shot visual reasoning, effectively combining the flexibility of neural pattern recognition with the precision, interpretability, and rigor of symbolic systems. By formalizing human conceptualization through the Computational Conceptualization framework and learning transformation primitives from data, *MIXAL* applies these through a sophisticated memory-augmented iterative refinement process. This innovative integration directly addresses fundamental limitations of current *ARC* reasoning systems, particularly the symbol grounding problem and context loss in multi-step reasoning.

The architecture's core emphasis on full interpretability, systematic exploration of the hypothesis space, and its alignment with human-like problem-solving strategies positions *MIXAL* as not only a highly competitive solution for *ARC* but also as a profound research contribution to the broader challenge of building AI systems capable of flexible, few-shot generalization and genuine understanding.

Our approach robustly demonstrates that the apparent complexity of *ARC* tasks can be systematically decomposed into learnable, composable, and grounded primitives. This is achieved by adhering to the critical insight that these tasks must ultimately remain solvable by human cognition, implying an underlying structure that our system can discover and exploit. This fundamental insight, coupled with our technical innovations, may prove invaluable for advancing artificial general intelligence research beyond the specific domain of visual reasoning into broader cognitive capabilities.

Acknowledgments

We acknowledge the *ARC* Prize organizers for creating this important benchmark and the research community for advancing our understanding of few-shot reasoning and neural-symbolic integration.

References

[References would be included in a full academic paper, covering relevant work in neural-symbolic systems, few-shot learning, memory-augmented networks, and *ARC* reasoning approaches]