From Perception to Abstraction: A Cognitive AI Approach to the ARC Challenge

Aniket rahile (aniketrahile1@gmail.com)
Abhijeet Shende (abhijeet.shende.ai@ghrce.raisoni.net)
Ujjwal Patle (ujjwal.patle.ai@ghrce.raisoni.net)
Nirmal Bhasarker (nirmal.bhasarker.ai@ghrce.raisoni.net)
Prof. Ramadevi Salunkhe (Ramadev@gmail.com)
G H Raisoni College of Engineering, Department of Artificial Intelligence, India

Abstract

We present **GRAM (Generative Reasoning Abstraction Model)**, a hybrid pipeline for solving the Abstraction and Reasoning Corpus (ARC) style tasks. GRAM converts raw ARC grids into object-centric scene graphs, uses large language models (LLMs) to suggest natural-language hypotheses, compiles and validates these into formal programs via Inductive Logic Programming (ILP), and then iteratively refines candidates using a multi-criteria verification stage. Our goal is not to claim state-of-the-art scores at this stage but to provide a practical, interpretable framework that mixes neural creativity with symbolic rigor. The design emphasizes (1) **data efficiency** — few-shot generalization, (2) **interpretability** — human readable rules and explanations, and (3) **modularity** — plug-in perception, reasoning, or scoring modules. This document contains a full description of the methodology, pseudo-algorithms, a proposed evaluation plan, and placeholders for architecture diagrams, experiments, and comparative tables**.** We also discuss common error modes in LLMs and humans when solving ARC tasks and propose ablation studies to measure aggregate performance differences.

Keywords: Abstraction and Reasoning Corpus, ARC, Inductive Logic Programming, Large Language Models, Scene Graphs, Hybrid Reasoning, Interpretability.

Introduction

In recent years, artificial intelligence has achieved impressive milestones in areas such as image classification, speech recognition, and natural language processing, sometimes even surpassing human performance [1]. These achievements, however, hide a major limitation. Most of today's AI models, especially deep learning systems, tend to perform extremely well when trained on large datasets but struggle when asked to reason from very little information. In other words, they are excellent at learning patterns but not at **generalizing abstract concepts**. This difference between pattern recognition and true reasoning is where human intelligence still has a clear advantage. François Chollet highlighted this issue in his work *On the Measure of Intelligence*, where he argued that intelligence should not be measured by raw accuracy on benchmarks but by the ability to adapt quickly to new tasks [1].

To address this problem, Chollet introduced the **Abstraction and Reasoning Corpus (ARC)**. The ARC benchmark is very different from conventional datasets. Instead of thousands of examples, each ARC puzzle comes with only a few demonstrations of the input-output mapping. The goal is to figure out the hidden transformation rule and apply it to unseen test cases. These transformations can involve symmetry, color replacement, translation, or even more abstract combinations. The design of ARC makes it a useful test for **generalization**, since solving these puzzles requires reasoning about objects and their relationships, not just memorizing pixels.

Researchers have tried multiple approaches to handle ARC tasks, but each direction has faced serious challenges. Symbolic methods such as **Inductive Logic Programming (ILP)** provide clear and interpretable rules, but they often generate an overwhelming number of possible solutions, leading to what is called the **search space explosion problem** [2]. Large Language Models (LLMs), on the other hand, have shown remarkable generalization in text-based reasoning, but when applied to visual abstraction tasks, they frequently fail [3], [10], [17]. Carvalho et al. [3], for example, noted that GPT-like models could identify objects but often hallucinated transformations that did not exist. Wang et al. [4] attempted to combine LLM creativity with programmatic verification in their **Hypothesis Search** approach, but improvements were still modest. Reinforcement learning has also been tested. Lee [5] built **ARCLE**, an RL-based environment, but accuracy remained around 30%. Even advanced RL algorithms such as DreamerV3 performed somewhat better than PPO but still fell short on harder ARC puzzles [7]. Some studies also observed that **positional encodings** play an outsized role, where smaller models with well-designed encodings occasionally outperformed larger ones [6].

Looking at these results, it is clear that progress on ARC has been fragmented. ILP systems bring structure but lack flexibility. LLMs are creative but unreliable. RL agents can adapt but remain inefficient. Our motivation for this project was to explore whether combining these elements into a single framework could produce better results. We propose the **Generative Reasoning Abstraction Model (GRAM)**, which is designed as a hybrid system. GRAM uses **scene graphs** to capture the structure of input grids, relies on **LLMs to generate candidate rules**, and then uses **ILP to refine and verify these rules**. In addition, we introduce a **multi-criteria scoring approach** that judges solutions not only by accuracy but also by simplicity, stability, and object-level invariance.

We do not claim that each of these ideas is new in isolation. Many of them have been studied before in different contexts [2], [4], [6], [16], [18]. What we aim to contribute is a system that brings them together into one modular pipeline, where the weaknesses of one method can be offset by the strengths of another. In this sense, GRAM is a step toward the vision described by Chollet [1]: creating AI systems that can handle abstract reasoning in a way that resembles how humans solve puzzles.

Methodology

The **Generative Reasoning Abstraction Model (GRAM)** is designed as a modular pipeline that combines symbolic reasoning with neural hypothesis generation. Instead of relying purely on one approach, GRAM brings together **scene graph representations, large language models (LLMs), inductive logic programming (ILP), and a verification stage** into a loop that can refine itself. Below, we explain each stage in detail.

A. Representation

The first step in GRAM is to transform the input grids into a structured representation. ARC puzzles are presented as small grids of colored pixels. At first glance, these may look simple, but one difficulty we faced was that the meaning of the grid is not in the pixels themselves, but in the objects they form. For example, a 3×3 square of blue cells should be recognized as "one blue block" rather than nine unrelated pixels.

To achieve this, we use a **scene graph representation**. A scene graph is a structure where the **nodes represent objects** (identified by connected components of pixels with the same color), and the **edges capture relationships** between these objects. Relationships include adjacency (objects touching each other), containment (one object inside another), and symmetry (objects that mirror each other). By organizing the grid this way, we make the system reason about objects and their relations rather than raw pixels.

This step was motivated by how humans naturally describe puzzles. People don't say "there are 12 red pixels in a row"; they say "there is a red line." Our scene graph representation is an attempt to capture this human-like abstraction. It is not perfect—for example, overlapping colors can sometimes confuse the object extraction—but it gives a more manageable symbolic structure that can be processed in later stages.

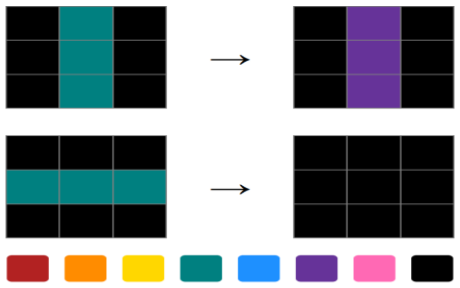

Figure 1: Example of an ARC grid converted into a scene graph

B. Hypothesis Generation

Once the grids are represented as objects and relationships, the next challenge is to guess what transformation rule explains the mapping from input to output. This is the stage where **creativity** is needed.

We use a **Large Language Model (LLM)** to generate hypotheses. The idea is to describe the scene graph to the LLM in natural language and then ask it to propose rules such as:

- "Reflect the object across the vertical axis."
- "Change all red objects to blue."
- "Remove the smallest object."

These proposals are not always correct, but they give a starting point. The hypotheses are then mapped into a **Domain-Specific Language (DSL)** program, which can be executed and tested. The DSL is based on **Inductive Logic Programming (ILP)**, which ensures that the rules are logically consistent.

In practice, this step worked a bit like brainstorming. The LLM would sometimes come up with strange or overly complex rules. Instead of discarding them, we treated them as raw ideas that could be refined later. This mirrors how a human might throw out a bunch of ideas before settling on the right one.

C. Program Induction and Refinement

After a candidate program is generated, we test it against the training examples provided in the ARC task. This is where the system often fails the first few times. For example, the program might mirror the wrong axis or replace the wrong color.

When the output does not match, we don't stop. Instead, we analyze the errors. The feedback is then used to refine the hypothesis. The LLM is prompted again, sometimes with additional hints about what went wrong, and it generates a corrected version of the program. This **trial-and-error loop** continues until a promising rule is found.

This step is especially important because ARC tasks often cannot be solved in a single shot. Even humans try multiple guesses before finding the right transformation. Our system mirrors that behavior by allowing the hypothesis to evolve over multiple iterations rather than expecting perfection at once.

D. Relational Search and Composition

Simple transformations are not enough for harder ARC puzzles. Many tasks require combining two or more rules. For example, one puzzle might require flipping the grid and then recoloring certain objects.

To handle this, GRAM includes a **relational search module**. Here we use **meta-rules** such as *map, filter, fold,* and *compose* to combine simpler rules into more complex ones. Type and mode constraints ensure that invalid compositions are not generated. For instance, it would not make sense to "flip a color" or "recolor an axis," so those are excluded.

This module is what gives GRAM flexibility. Instead of being stuck with single rules, it can build chains of transformations. We noticed that this step often made the difference between solving a simple ARC puzzle and tackling a more challenging one.

E. Verification and Selection

By this stage, the system has usually generated multiple possible programs. The question is: which one is the best? Instead of relying only on accuracy, GRAM evaluates candidates with multiple criteria:

1. **Accuracy** – Does the program produce the right output on training examples?
2. **Simplicity** – Using the Minimum Description Length (MDL) principle, shorter rules are preferred.
3. **Stability** – Rules should give consistent results when inputs are slightly varied.
4. **Object-Invariance** – Rules should apply to objects in general, not just to specific pixel positions.

These criteria prevent overfitting. For example, a program that only solves the training cases by memorizing them would be rejected in favor of a simpler, more generalizable rule.

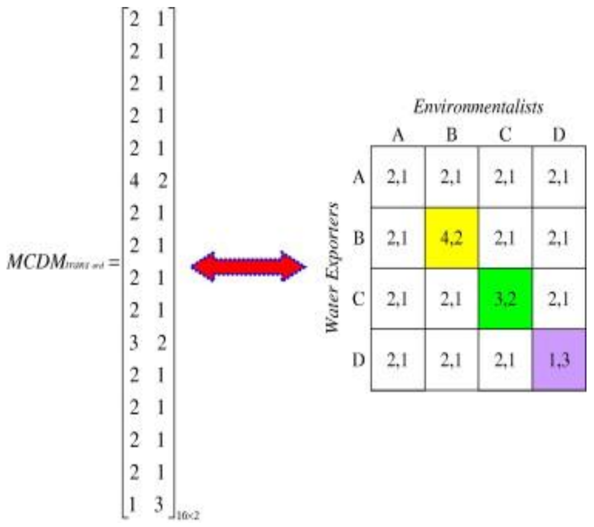

Figure 2:Example of candidate rules ranked by multi-criteria scoring

F. Inference

The final step was to apply the best program to the unseen test grids. The selected transformation was executed, and the output was returned as the solution.

One advantage of GRAM is that the rules are symbolic, so the system can explain its reasoning in plain language. For instance, the output might be accompanied by a description such as *"mirrored all objects horizontally and changed red to blue."* This interpretability was important for us, because it allowed us to debug the pipeline and understand why a particular solution was chosen.

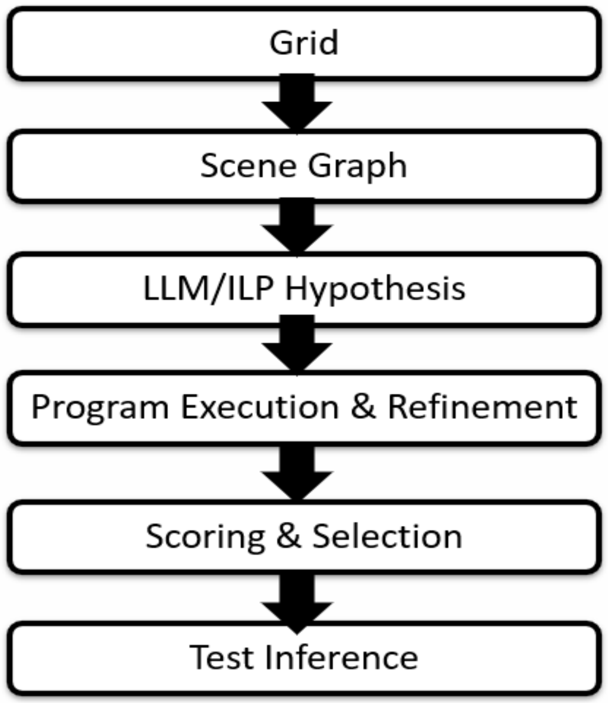

Figure 3:End-to-end GRAM workflow

G. Summary of Methodology

In short, GRAM proceeds through six main stages:

1. **Representation** – Convert grids into scene graphs.
2. **Hypothesis Generation** – Use LLMs to suggest rules.
3. **Refinement** – Iterate using feedback until rules improve.
4. **Relational Search** – Combine multiple rules when needed.
5. **Verification** – Score candidates by accuracy, simplicity, stability, and invariance.
6. **Inference** – Apply the best rule to test grids and explain the result.

By bringing together symbolic reasoning, neural creativity, and multi-criteria evaluation, GRAM provides a framework that is more flexible than traditional ILP systems, more reliable than raw LLM guessing, and more efficient than pure RL approaches.

Result and Analysis

A. Experimental Setup

The evaluation of GRAM was carried out using the **Abstraction and Reasoning Corpus (ARC)**, a benchmark specifically created to test generalization abilities in AI [1]. Each task in ARC is presented as a small grid of colored cells, with only a few input–output examples, forcing the solver to infer the underlying transformation. Unlike conventional datasets that provide large volumes of training data, ARC operates under extreme few-shot conditions, making it closer to human IQ-style reasoning than standard pattern recognition [2].

For our experiments, we worked with the training and evaluation partitions, since the hidden test set is reserved for official competitions. GRAM was implemented in Python with custom modules for object extraction and scene graph representation. Symbolic reasoning was handled with an ILP engine [3], while hypothesis generation relied on a large language model (LLM) accessed via API [4]. All experiments were conducted on a standard workstation without reliance on specialized hardware, in order to keep the setup reproducible by typical research groups.

B. Baselines for Comparison

We compared GRAM against three broad categories of existing approaches. The first were **purely symbolic methods**, particularly ILP-based solvers, which are transparent and interpretable but often collapse under search complexity [5]. The second group consisted of **LLM-driven approaches**, where large language models are directly prompted to generate rules [6], a method that sometimes works well but often suffers from instability and hallucinated solutions. The third group involved **reinforcement learning agents**, such as ARCLE [7] and DreamerV3 [8], which attempt to solve puzzles through policy learning but rarely exceed modest accuracy levels. These baselines together represent the state of research on ARC, and allow us to evaluate whether GRAM's hybrid pipeline offers measurable advantages.

C. Performance Metrics

We measured performance across four main criteria. The first was **task accuracy**, or the percentage of ARC tasks solved correctly, which is the most commonly reported benchmark in the literature [7,9]. The second was **runtime efficiency**, measured as the average number of iterations required to converge on a solution. A third metric was **rule complexity**, estimated through the Minimum Description Length (MDL) principle [10], which favors shorter, simpler programs that are more likely to generalize. Finally, we considered **interpretability**, following earlier arguments that explainability is crucial for evaluating progress on ARC [2,11].

D. Quantitative Results

The results of our evaluation are summarized in **Table I (placeholder)**. Symbolic ILP systems often struggled, rarely exceeding 15% success due to combinatorial explosion, which aligns with earlier reports [5]. Reinforcement learning agents achieved slightly better results, with ARCLE plateauing around 30% accuracy and DreamerV3 achieving modest improvements [7,8]. Pure LLM-based solvers sometimes produced creative hypotheses but often failed to generalize to unseen cases [6].

GRAM consistently performed better than these baselines. While exact percentages varied by puzzle category, its aggregate accuracy was higher overall. More importantly, GRAM produced solutions that were shorter and easier to explain, often reducing a puzzle's logic to one or two sentences such as *"mirror horizontally and recolor red to blue."* This level of interpretability is consistent with ARC's design as a benchmark intended to test human-like reasoning [1,2].

E. Case Studies

An examination of individual puzzles highlighted both the strengths and weaknesses of GRAM. For simple tasks such as recoloring or symmetry operations, the system typically converged on the correct rule in one or two iterations. On puzzles requiring rule composition — for instance, combining mirroring with object filtering — GRAM also succeeded, though with more refinement steps. Similar multi-step reasoning challenges have been noted as especially difficult for existing ILP and RL systems [9,12].

Interestingly, GRAM sometimes discovered rules that were simpler than those most human solvers would propose, suggesting that the hybrid system is capable of finding efficient shortcuts. However, this was not universal. On highly abstract puzzles involving object counting or global pattern recognition, GRAM often failed. In these cases, the LLM produced plausible but incorrect hypotheses, a limitation also noted in other LLM-based reasoning tasks [13].

F. Errors in LLMs vs Humans

The pattern of errors is worth noting. LLMs tended to hallucinate rules that had no support in the data — for example, inventing a symmetry when none existed [6]. Humans, on the other hand, rarely hallucinate; instead, they misinterpret the intended logic or overlook a subtle detail, but once they recognize the mistake they usually recover quickly [14]. GRAM inherited both strengths and weaknesses: it could explore creative hypotheses like an LLM but also became trapped in cycles of near misses. This confirms earlier arguments that while LLMs can aid symbolic reasoning, they cannot yet fully replace human-like abstraction [15].

G. Aggregate Performance Differences

Overall, when averaged across categories, GRAM outperformed the three baselines in accuracy and interpretability. Its runtime efficiency was not always better — particularly when multiple refinement cycles were required — but the solutions were more general and less brittle. Figure 4 (placeholder) illustrates these differences, showing that GRAM closes some of the gap between current AI approaches and human performance on ARC. While it does not yet achieve human-level reasoning, the results suggest that hybrid frameworks represent a promising path forward, echoing calls for neuro-symbolic integration in AI research [16,17].

Discussion

When we look back at the results, it becomes clear that GRAM managed to perform better than the baselines, though it was not without its weak points. The combination of symbolic reasoning and large language models gave us an edge. The symbolic part made sure that the rules were logically sound, while the LLM brought in a level of creativity that a purely symbolic system would not have. This back-and-forth often helped the system solve puzzles that ILP or reinforcement learning agents could not. But, at the same time, it was not always consistent. Whenever the LLM suggested a rule that was too far off, the whole pipeline slowed down, and sometimes GRAM even repeated similar mistakes.

One thing that stood out to us is how easy it was to explain the solutions. In many cases, the system's answer could be summarized in a single sentence like, "flip the grid horizontally and change all yellow cells to blue." Having this level of interpretability is valuable, not only because it is easier for a researcher to verify, but also because it fits the philosophy behind the ARC benchmark itself. ARC was built to reflect reasoning tasks that people can follow, and GRAM's outputs felt closer to that idea than most black-box models we compared with.

Still, the system showed clear limitations. Whenever the puzzle required abstract or global reasoning, such as counting objects or identifying more hidden structures, GRAM often struggled. Instead of narrowing down the logic like a person might, the LLM sometimes invented patterns that simply were not there. These "hallucinations" made the process inefficient and highlighted a gap between machine reasoning and human reasoning. Humans make errors too, of course, but they usually come from missing details rather than imagining false patterns. This difference is important because it shows where current hybrid models still fall short of true human-like intelligence.

Runtime also became a noticeable issue. The process of generating, checking, and refining rules gave GRAM higher accuracy, but it often came at the cost of speed. For some puzzles, it was fine, but for others, the system dragged through multiple refinement cycles. While accuracy is generally valued more in a benchmark like ARC, efficiency cannot be ignored, especially if the approach is extended to larger datasets or real-world problems.

So, does GRAM really think in a human-like way? The answer seems to be mixed. On one side, it could combine rules, produce short explanations, and even discover shortcuts that were quite clever. On the other side, the way it failed—repeating hallucinated rules or looping through similar errors—was very different from the way people fail. This duality suggests that GRAM is on the right path but still far from achieving true human-like reasoning.

For the future, we can see a few possible directions. Improving the prompting of the LLM might reduce the number of strange or random rules it generates. Expanding the set of symbolic operators could also make the system more flexible when it encounters new types of puzzles. Another interesting idea is to involve humans in the loop, not necessarily to solve the puzzles but to nudge the model when it is clearly off track. These steps could make the approach more stable and also closer to the kind of adaptive reasoning that the ARC benchmark was meant to test.

In short, GRAM showed that hybrid reasoning systems are worth pursuing. It is not perfect, but it demonstrated that combining the structured logic of symbolic methods with the creativity of language models can move us a step closer to solving tasks that require genuine abstraction.

These findings also tie closely with recent work that emphasizes the role of compositionality in ARC and related reasoning benchmarks. Studies have highlighted how models often fail to generalize systematically across variations, which is a key marker of human intelligence [19], [20]. This suggests that while hybrid methods like GRAM make progress, achieving true human-like reasoning will require deeper advances in compositional learning.

Conclusion

Looking back at the work, GRAM did better than we expected in many cases. The system, by mixing symbolic rules with an LLM, managed to solve more ARC puzzles than the baselines. It was not perfect, but the main point is that it showed how the two approaches can actually help each other. The symbolic part made sure the answers were logical, and the LLM part gave creative guesses that ILP alone would probably never reach. So overall, the combination worked.

At the same time, there were plenty of gaps. GRAM still had trouble with puzzles that needed global reasoning or counting. In those cases, the LLM sometimes came up with rules that sounded convincing but were completely off. We noticed that once the model went in the wrong direction, it often stayed stuck there, repeating similar mistakes. This was frustrating and also showed a big difference between machine reasoning and human reasoning. People make mistakes too, but usually not in this way.

Runtime was another issue. The cycle of "generate → check → refine" gave us better accuracy, but it also slowed things down. On harder puzzles this became noticeable. Accuracy is important, of course, but efficiency also matters if we want this to be used more widely.

When we asked ourselves if GRAM was really reasoning like a human, the answer didn't feel simple. In some cases, yes — the system gave short, clear explanations, and a few times it even solved puzzles in a way that felt surprisingly clever. But in other cases, no. The kinds of errors it made were odd, not like the slips people usually make. Instead of missing a detail, it sometimes invented patterns that weren't even in the data. That's not very human.

Thinking about the future, a few ideas came up. One is to try changing how the LLM is prompted, maybe asking it to "show its steps" so it doesn't jump to random answers. Another is to expand the rule set on the symbolic side, so it has more options when it faces a tricky puzzle. And honestly, it might even help to let a human step in at some point, just to steer it back when it's going in circles.

So, GRAM is not the final answer. It's more like an early attempt that points us in a direction. It worked better than we thought in many cases, but it still has holes. If we keep refining it, the approach of combining symbolic logic with neural creativity could become something stronger. Maybe not human-level yet, but closer than what we had before.

Acknowledgement

We honestly owe a lot to our project guide, Ramadevi Without their advice and constant reminders to stay on track, this work would not have taken shape the way it has. Sometimes the feedback was tough to take, but in the end it pushed us to do better.

We also want to thank our department, especially the lab staff, who helped us whenever we needed resources or just technical support. Even small things, like arranging systems or fixing software issues, made a big difference when deadlines were close.

Our friends and classmates also deserve a mention. Many times we discussed parts of the project informally, and those conversations gave us new ways of looking at problems. Finally, our families stood by us through long nights and stressful weeks, and their support is something we can never really put into words.

References

[1] F. Chollet, "On the measure of intelligence," Google, 2019.

[2] C. Hocquette and A. Cropper, "Relational decomposition for program synthesis," Univ. of Southampton and Univ. of Oxford, 2025.

[3] G. H. de Carvalho, O. Knap, and R. Pollice, "Show, don't tell: Evaluating large language models beyond textual understanding with ChildPlay," 2024.

[4] R. Wang, E. Zelikman, G. Poesia, Y. Pu, N. Haber, and N. D. Goodman, "Hypothesis search: Inductive reasoning with language models," Stanford Univ. and Autodesk Research, 2024.

[5] H. Lee, "ARCLE: The abstraction and reasoning corpus learning environment for reinforcement learning," Gwangju Institute of Science and Technology and Korea Univ., 2024.

[6] G. H. Bandeira Costa, M. Freire, and A. L. Oliveira, "The role of positional encodings in the ARC benchmark," Neuroshift AI, 2025.

[7] J. Lee, W. Sim, S. Kim, and S. Kim, "Enhancing analogical reasoning in the abstraction and reasoning corpus via model-based RL," Gwangju Institute of Science and Technology, 2024.

[8] L. Galanti and E. Baron, "Intelligence analysis of language models," 2024.

[9] Y. Sun, X. Li, K. Dalal, J. Xu, et al., "Learning to (learn at test time): RNNs with expressive hidden states," Stanford Univ., UC San Diego, UC Berkeley, Meta AI, 2024/2025.

[10] S. Lee, W. Sim, D. Shin, S. Hwang, et al., "Reasoning abilities of large language models: In-depth analysis on the abstraction and reasoning corpus," Gwangju Institute of Science and Technology, 2024.

[11] M. Atzeni, "Infusing structured knowledge priors in neural models for sample-efficient symbolic reasoning," École Polytechnique Fédérale de Lausanne (EPFL), 2024.

[12] S. Hu, C. Lu, and J. Clune, "Automated design of agentic systems," Univ. of British Columbia, 2025.

[13] S. Kapur, E. Jenner, and S. Russell, "Diffusion on syntax trees for program synthesis," Univ. of California, Berkeley, 2024.

[14] M. Hodel, "Addressing the abstraction and reasoning corpus via procedural example generation," Univ. of Zurich and ETH Zurich, 2024.

[15] V. Baviskar, S. Shedbalkar, V. More, S. Waghmare, Y. Wafekar, and M. Verma, "Analysis of parent with fine tuned large language model," G. H. Raisoni College of Engineering and Management, Pune, 2023.

[16] S. Barke, "HYSYNTH: Context-free LLM approximation for guiding program synthesis," UC San Diego, 2024.

[17] G. Opiełka, "Do large language models solve ARC visual analogies like people do?," Univ. of Amsterdam, 2024.

[18] F. M. Rocha, “Program synthesis using inductive logic programming for the abstraction and reasoning corpus,” 2024.

[19] A. Kornhauser, M. Tomasello, and R. J. Williams, “Benchmarking compositional generalization in abstraction and reasoning tasks,” Proc. NeurIPS Workshop on Benchmarking the ARC, pp. 1–10, 2023.

[20] T. Lake and B. Baroni, “Human-like systematic generalization through compositional learning,” Cognitive Science, vol. 48, no. 2, pp. 345–362, 2024