ARCANA: A Benchmark for Abstraction and Analogical Reasoning

Ruqi Chen
University of Science and Technology of China
Hefei, China
State Key Laboratory of General Artificial Intelligence, BIGAI
Beijing, China
crq2020_08@mail.ustc.edu.cn

Zilei Wang*
University of Science and Technology of China
Hefei, China
zlwang@ustc.edu.cn

Wei Wang*
University of Science and Technology Beijing
State Key Laboratory of General Artificial Intelligence, BIGAI
Beijing, China
wangwei@nlpr.ia.ac.cn

Abstract

Analogical reasoning is widely considered a fundamental component of human intelligence. Few benchmarks are designed to explicitly test far analogical reasoning—reasoning between domains that are superficially dissimilar yet share deep relational or structural similarities. In this work, we introduce an analogical reasoning benchmark called ARCANA. The tasks of ARCANA require performing far analogical mappings from real-world objects or physical phenomena to abstract grid-based transformations, different from the well-known ARC-AGI benchmark which primarily emphasized innate human priors. We validate ARCANA through a human study, demonstrating that humans solve these tasks reliably. In contrast, evaluations across a range of large language models show a distinct performance gap compared to humans, revealing systematic failure modes that remain challenging for current models. To address this gap, we propose CONCEPTSIR, a reasoning framework based on concept slippage, introspection, and reflection. Experiments demonstrate that CONCEPTSIR enables standard LLMs to solve several challenging ARCANA tasks that were unsolved by direct prompting in our evaluation. Overall, our results suggest that far analogical reasoning remains an open problem for contemporary LLMs. All the materials are available at sites.google.com/view/arcanacogtest. The complete codebase, including benchmark, and ConceptSIR, is publicly available at `https://github.com/density-11/arcana-test`. All ARCANA data and task instances can be accessed and explored at `https://arcanacogtest.org/`.

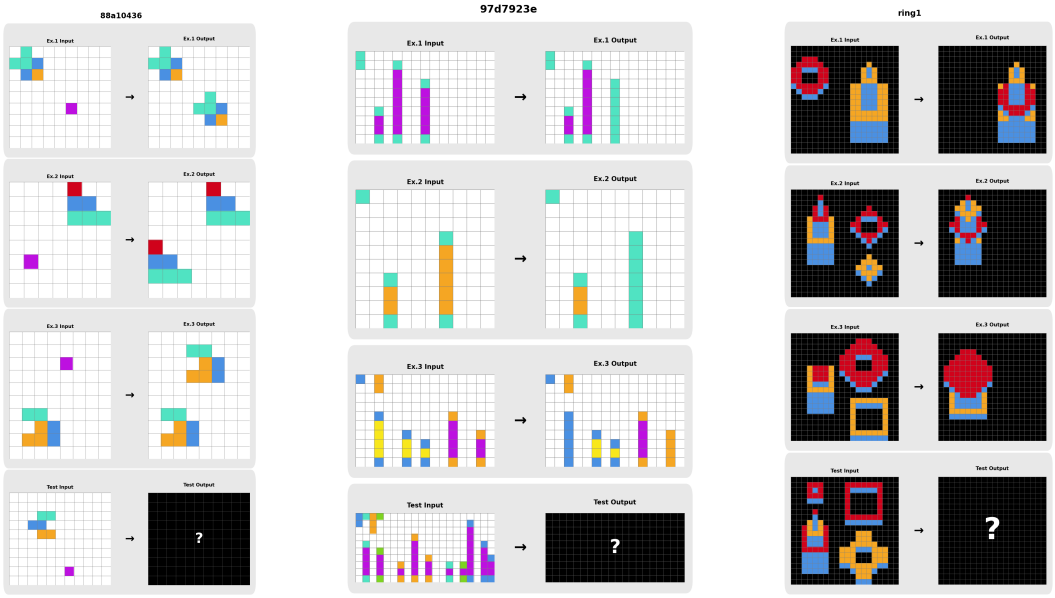
*Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Analogical reasoning has long been regarded as a central component of human intelligence and a characteristic of general intelligence more broadly Gentner [1983], Hofstadter et al. [2001, 1995], Holyoak and Thagard [1989]. Humans routinely draw analogies between domains that are superficially dissimilar yet share deep relational structure, allowing them to generalize knowledge flexibly from limited experience. This capability for far analogy—mapping abstract relations across distant domains—plays a critical role in human learning, reasoning, and problem solving.

In recent years, substantial progress has been made in evaluating the general reasoning abilities of artificial intelligence systems, such as Humanity's Last Exam Phan et al. [2025] and SWE-Bench Pro Deng et al. [2025]. However, these benchmarks are not designed to explicitly probe far analogical reasoning. Among the most challenging existing benchmarks, ARC-AGI-1&2 evaluate abstraction, reasoning, and analogy through grid-based tasks that require inferring latent rules from a small number of examples Chollet et al. [2025]. In general, ARC-AGI-1&2 primarily focus on testing near analogy based on relative surface-level feature similarity (e.g., color, shape, position, number, center, copy, and count). While instances of far analogy may arise implicitly, explicit far analogical reasoning—particularly that requiring mappings from **real-world concepts or physical phenomena to abstract symbolic representations**—is rare and limited in scope. This leaves a substantial gap in the systematic evaluation of fluid intelligence and far analogical reasoning.



(a) ARC-AGI-1 (b) ARC-AGI-2 (c) ARCANA

Figure 1: Visualization of three tasks from ARC-AGI-1, ARC-AGI-2, and ARCANA. (a) Copy the 3×3 pattern to the purple block which is the center of the destination. (b) Fill in the columns with different colors, and the block numbers of the same color in the upper left corner determine the columns to be filled. (c) ARCANA focuses on far analogical reasoning involving threading 3D rings onto a column, where the rings settle at equilibrium positions determined by the geometry of their inner diameters.

To address this gap, we introduce ARCANA (Abstraction and Reasoning Corpus for ANAlogy), a benchmark explicitly designed to evaluate far analogical reasoning. As shown in Figure 1, unlike ARC-AGI-1&2 which largely emphasizes innate human priors (e.g., copying the 3×3 pattern to the purple grid, filling in the grids with different colors using discovered rules), ARCANA focuses on far analogical mappings from real-world objects or physical phenomena to ARC grid transformations (e.g., threading 3D rings onto a column with physical constraints). The benchmark draws from a broad and diverse set of conceptual sources, including physical principles (e.g., buoyancy, levers, gears, melting, the Doppler effect, Kepler's second law), natural phenomena (e.g., planet, sunsets), and object-level abstractions (e.g., cats, ducks, fireworks, 3D rings, 3D sushi).

To assess the validity and difficulty of ARCANA tasks, we conduct a human study with undergraduate participants. The results show that humans reliably solve ARCANA tasks, indicating that the benchmark captures forms of reasoning that are natural for human cognition. Meanwhile, we evaluate ARCANA across a range of large language models (LLMs), the results of which reveal lower performances, suggesting that far analogical reasoning remains a challenge for current models and exposing systematic failure modes beyond simple pattern matching.

Inspired by the Copycat model of analogy making proposed by Hofstadter et al. [1995], we propose CONCEPTSIR, an analogical reasoning framework based on concept slippage, introspection, and reflection. CONCEPTSIR is designed to approximate key aspects of the human problem-solving process when constructing and solving ARCANA tasks. Empirical results show that CONCEPTSIR enables large language models to solve several ARCANA tasks that are otherwise unsolved by standard prompting strategies, providing insight into the potential of our CONCEPTSIR.

In summary, this work makes three contributions: (1) we introduce ARCANA, a benchmark explicitly targeting far analogical reasoning; (2) we present a human and model evaluation that reveals a substantial performance gap; and (3) we propose CONCEPTSIR, a reasoning framework that improves model performance while highlighting remaining challenges. Together, our results suggest that far analogy remains an open problem for contemporary AI systems and that the reasoning pipeline of CONCEPTSIR offers a promising direction for studying analogical reasoning.

2 Related Work

2.1 Cognitive Theories of Analogical Reasoning

Cognitive theories have long sought to formalize analogy-making. Structure-Mapping Theory (SMT) views analogy as source–target mapping that preserves relational structure rather than surface features Gentner [1983]; SAGE extends SMT with frequency-based generalization over alignable structures Forbus et al. [2017]; and ACME models analogical mapping as constraint satisfaction Holyoak and Thagard [1989]. Our CONCEPTSIR is most directly inspired by Copycat Hofstadter et al. [1995, 2001], which uses a Slipnet, Workspace, and perceptual agents to model flexible concept slippage in analogy-making.

2.2 Analogical Reasoning Benchmarks and Models

Analogical reasoning has been evaluated through cognitive tests and benchmarks such as Raven's Progressive Matrices (RPM), a canonical measure of human fluid intelligence John and Raven [2003], Zhang et al. [2019], and the Abstraction and Reasoning Corpus (ARC), which emphasizes abstraction and analogy Chollet [2019], Chollet et al. [2025]. ARC-AGI-2 further increases difficulty through more abstract conceptual compositions Chollet et al. [2025]. However, benchmarks for far analogical reasoning grounded in real-world concepts remain limited, motivating ARCANA.

For modeling, LLMs have become a major approach to ARC-AGI-1&2 due to their broad pretraining and general reasoning ability. Other ARC-oriented methods include train-from-scratch ViT-based picture-to-picture mappings Hu et al. [2025] and Minimal Description Length (MDL)-inspired approaches Liao and Gu [2025]. Beyond ARC, prior work includes vector-based matching with BART embeddings Lu et al. [2019] and encoder–decoder models combined with Structure-Mapping Theory to induce structured correspondences Crouse et al. [2021].

3 ARCANA: the Abstraction and Reasoning Corpus for ANAlogy

3.1 Dataset Creation

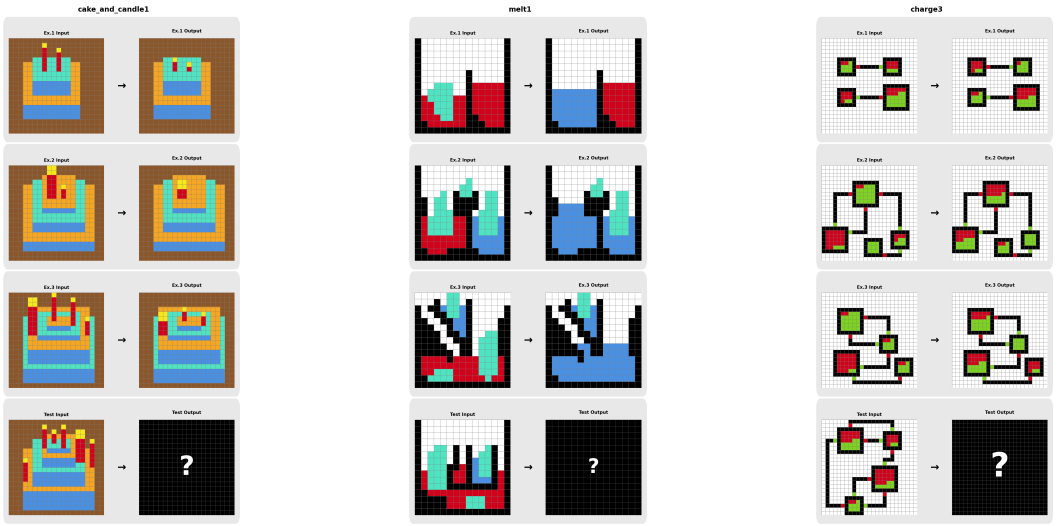
3.1.1 Target-concept-target generation loop

ARCANA tasks are grounded in diverse real-world objects, concepts, and physical phenomena, which we refer to as **targets**. Each target can be abstracted into a small set of higher-level **concepts** that govern the grid transformation. For instance, the "threaded rings" target (shown in Figure 1c) can be summarized by three concepts—move, shelter, and 3D—where 3D rings move and partially occlude ("shelter") the column pattern. Once such concepts are extracted, they can be slipped and

re-grounded to other targets (e.g., candles burning on a cake, as shown in Figure 5a). With this target-concept-target generation loop, we create several different targets that nevertheless instantiate the same underlying rule on ARC grids.

Although some tasks in ARCANA might be mapped from physical phenomena, ARCANA still holds a crucial difference from physical reasoning benchmarks. Many tasks in ARCANA are non-physical, including `cat1`, `shoes1`, etc. Even physical inspired tasks are highly abstracted away from real physics: `melt1`, `melt2`, `melt4` encode conservation of colored cells and a horizontal surface rather than density, or strict fluid dynamics; `buoyancy2` concerns grid relations rather than buoyant-force calculation; and even `Kepler_second_law_or_area1` can be solved through a grid invariant without knowing Kepler's law. Therefore, we address that ARCANA evaluates explicit transfer from real-world concepts to abstract grids, and physical phenomena are only one possible source of analogy; the source concept should not be confused with the capability being evaluated.

The target-concept-target generation loop can be viewed as an explicit process of analogy-making. We source targets from diverse domains (e.g., water, windmills, flags, and physical laws) and classify tasks into 34 concept categories plus an "other" class. Example concepts include wave propagation, 3D cuboid motion and shadows, fluid flow, state transitions, object-induced rotation, light propagation, and game-rule dynamics. Many tasks are compositional, combining multiple concepts (e.g., motion with rotation).



(a) Cake and candle (b) Melt (c) Charge

Figure 2: Visualization of three ARCANA task instantiations with distinct real-world targets and abstract concepts. (a) **Candles on a cake:** candles with different cross-sectional areas and lengths are placed on a 3D cake; after partial burning, they reveal the cake behind them. (b) **Melting:** teal cells are ice, blue cells are cold water, and red cells are hot water; ice melts only when touching hot water, and the melted ice plus hot water becomes cold water with volume preserved. (c) **Charge/data transfer:** green cells propagate along directed wires from the source container connected to the red terminal to the sink container connected to the green terminal; each wire transfers one cell per time step if the destination has capacity.

3.1.2 Grounding targets into grids

Concept attention. Bridging targets and discrete ARC grids requires concept attention: emphasizing task-relevant concepts of a target while suppressing concepts that are secondary or irrelevant to the core rule. For example, in the *melt* target (shown in Figure 5b), ice melts only upon contact with hot water (red cells); after melting, both the ice and the hot-water become cold water (blue cells). The final cold-water volume (the number of grid cells) equals the sum of the initial ice volume and hot-water volume. We therefore focus on the melting condition and this discrete conservation rule, while ignoring physical nuances such as density differences across phases and temperatures.

Concept expansion. Beyond concept attention, we sometimes augment a target with additional concepts (e.g., time) to fully specify its dynamics. We refer to this as concept expansion. For instance, in the *charge* target (shown in Figure 2c), green cells (analogous to charge or data) propagate along directed "wires" from a source container at the red terminal to a sink container at the green terminal. At each time step, each wire transfers one cell if the sink has remaining capacity. Introducing an explicit temporal process (and treating it as effectively continuous) is necessary to correctly solve the test. Overall, concept expansion substantially increases the difficulty and diversity of ARCANA tasks.

Generation principles We construct the benchmark according to four principles that are intentionally simple and human-grounded: (1) **Human-familiar targets.** The targets are common and recognizable to humans; given the target, most people who are familiar with it can solve the task accurately with little effort. (2) **Concept-faithful grounding.** The ARC grids inherit the target's core concepts, and the grid-target analogy is intended to be explicit. (3) **Multiple targets, one rule.** A task may admit multiple plausible analogical targets, but they must agree on the same underlying transformation rule, such as claw and chopsticks. (4) **Human-aligned correctness.** When multiple rules are consistent with the examples, we choose the one most natural to humans, best aligned with everyday concepts, and lowest in cognitive burden.

ARCANA comprises 200 tasks spanning 35 distinct concepts, with each task targeting a unique analogical mapping. Additional details are provided in Appendix F.

4 Human Study on ARCANA

4.1 Experiment setup

Since solving an ARCANA task often benefits from recognizing the intended real-world target, unfamiliarity with the target can increase failure rates. To reduce such idiosyncratic variance and obtain a more reliable estimate, each task is randomly assigned to at least six participants. We recruited undergraduate participants via an official university online platform and completed the study on a custom-built website. In addition to final grid answers, we ask participants to (i) provide a brief one-sentence description of the inferred target, preferably in analogical terms, and (ii) rate the task's thinking and drawing difficulty on a five-level scale (very easy to very hard), enabling finer-grained analyses of human reasoning. To ensure data validity, participants must pass a tutorial and an attention check; submissions from participants who fail the attention check are excluded. After filtering, we obtain 1659 question-level responses from 80 participants. All details are provided in Appendix B.2.

The reported human accuracy may vary across populations, because performance on far-analogy tasks partly depends on familiarity with the targets. To reduce cultural variation, we primarily selected broadly recognizable targets and concepts, although it is not enough to guarantee equal familiarity across populations, even on the participants recruited. Our undergraduate sample was chosen to provide participants with a reasonable level of familiarity with the concepts represented in ARCANA. For example, lower performance among young children or participants unfamiliar with a physical law may reflect limited source-domain knowledge rather than difficulty with analogy itself. Conversely, these participants might perform well on far analogies grounded in domains more familiar to them.

Therefore, the reported accuracy should be interpreted in the context of the sampled undergraduate population and may not directly generalize to others. Nevertheless, the results provide evidence that, when the underlying targets are familiar, humans can successfully perform the type of far analogical mapping tested by ARCANA.

4.2 Experiment analysis

ARCANA is designed around human-centered priors, therefore human performance provides a natural reference point. Moreover, ARCANA tasks may admit multiple plausible targets that nonetheless instantiate the same underlying rule (Principle 3), automated evaluation could be unreliable; we therefore manually evaluate every response. All submissions are categorized into five types: **CC** (answer Correct, analogy Correct), **CW** (answer Correct, analogy Wrong), **WC** (answer Wrong, analogy Correct), **WW** (answer Wrong, analogy Wrong), and **WT** (answer Wrong with Typo). **WT**

captures minor typographical mistakes: the participant expresses the intended rule but leaves one or two erroneous cells/blocks that are typically irrelevant to the core transformation. Although typos exhibit several forms which makes systematic categorization challenging, we still provide a rough taxonomy and concrete examples in the Appendix B.5. If the mistakes affect the rule expression, we instead label the response as **WC**. The distribution of these categories is shown in Figure 3(a).

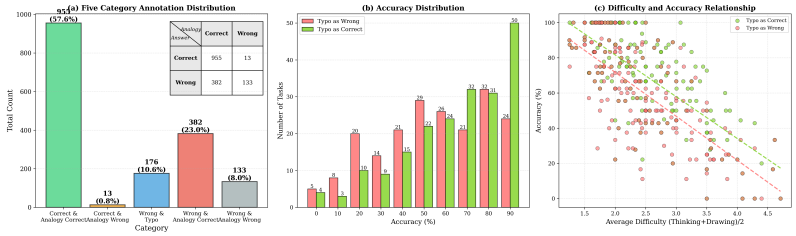


Figure 3: Human performance and subjective difficulty on ARCANA. (a) Response distribution over five manual categories (CC, CW, WT, WC, WW), with a 2×2 table summarizing answer vs. analogy correctness (typo excluded). (b) Per-task human accuracy histogram under two scoring conventions (typo counted as wrong vs. correct). (c) Per-task accuracy versus average subjective difficulty (thinking and drawing; mapped to a 1–5 scale), with fitted trend lines for both conventions.

We acknowledge that all results are evaluated manually by one author ensuring the same criteria were applied consistently. Nevertheless, this may possibly introduce subjective bias and precludes inter-rater agreement analysis. To make our evaluation more persuasive, we conservatively classify the ambiguous cases as incorrect and consistently report strict results without typo handling. Moreover, further analysis on Human (≥ 2) and average accuracy (in Appendix C.3) shows typo judgments do not drive the core conclusions.

To quantify the relationship between analogy and answer, we compute the Pearson coefficient ϕ on the 2×2 table after excluding typo responses. We obtain $\phi = 0.391$, indicating a moderate positive association between producing a valid analogical interpretation and producing a correct grid output. The fact that ϕ does not exceed 0.6 (i.e., not a strong association) is also unsurprising: even after a participant identifies a plausible analogy, successfully solving the task can still require additional concept expansion and careful, error-free operations on the grid. Overall, these results are sufficient to conclude that analogy and answers are meaningfully related, and that analogy can serve as a useful aid for solving ARCANA tasks.

We next analyze task-level human accuracy and subjective difficulty to assess whether ARCANA is human-aligned. Because typo responses are qualitatively distinct from genuine failures, we report two accuracy variants: counting typo as wrong versus as correct. Figure 3(b) shows that ARCANA exhibits a reasonable, non-trivial difficulty profile for humans: most tasks fall into intermediate-to-high accuracy ranges, and treating typo as correct shifts the distribution upward. Together, these results support the validity of ARCANA as a human-grounded benchmark. Figure 3(c) further relates accuracy to participants' subjective difficulty ratings, showing that most tasks are perceived as moderately difficult and that perceived difficulty is negatively correlated with accuracy. To quantify uncertainty in the human study, we compute 95% confidence intervals using a hierarchical bootstrap over tasks and human responses. The **task-level macro-average human accuracy** is 59.13% under strict evaluation (95% CI: 54.98%–63.31%) and 69.67% with typo tolerance (95% CI: 65.67%–73.65%).

Following ARC-AGI-2, we report Human (≥ 2) accuracy for ARCANA, defined as the proportion of tasks solved by at least two participants: 93.5% under strict evaluation and 96% under typo-tolerant evaluation, suggesting that most tasks are well-posed. Although filtering out tasks solved by fewer than two participants is reasonable, whether a task meets this threshold can be affected by participant variability. We therefore retain all tasks to preserve benchmark coverage and diversity.

More details are provided in the Appendix B. Overall, humans achieve 68.96% average accuracy on ARCANA, including 10.61% typo-level matches, while the Human (≥ 2) accuracy reaches 96%, including 2.5% typo-level matches. We further observe a moderate positive association between analogy correctness and answer correctness, suggesting that analogy provides a meaningful scaffold for solving ARCANA tasks. Finally, the resulting task-level accuracy and difficulty distributions

consistently indicate that ARCANA is broadly solvable for humans and aligned with human reasoning, supporting the validity of the benchmark.

5 LLMs Testing on ARCANA

5.1 Experiment setup

We evaluate all ARCANA tasks on four representative models: GPT 5.2, Gemini 3 Pro, Claude Sonnet 4.5, and Grok 4.1 Fast. These models are required to directly output the final answer grid, which is motivated by the fact that some ARCANA rules are difficult to faithfully express as executable code, making code-based evaluation potentially restrictive. Under this output format, LLMs may also produce minor typographical mistakes. Accordingly, all LLM responses are evaluated manually with the same criteria as in the human study, and results are reported under two conventions: counting typos as correct (`+Typo`) versus as wrong (`-Typo`). To further improve performance, we employ two prompting strategies: chain-of-thought and analogical prompting. Due to the substantial stochasticity in model outputs, we follow the common practice in ARC evaluations and report `pass@3` (top-3). All tests are finished through OpenRouter. Temperature was set to 1. Reasoning effort was not explicitly configured in our OpenRouter requests; therefore, each model used its model-specific default reasoning configuration. In the language-only setting, grids are serialized as numerical matrices with integers representing colors. In the vision-language setting, images are supplied through the standard image interface using Base64 encoding or URLs. Models output the final grid as a numerical matrix in both settings.

Model	ARCANA					
	Total		CoT (language)		CoT + Ana (VL)	
	+Typo	-Typo	+Typo	-Typo	+Typo	-Typo
GPT 5.2	**51.20%**	**48.33%**	**40.19%**	**38.28%**	**42.11%**	**37.32%**
Gemini 3 Pro	43.06%	29.19%	34.45%	23.92%	28.71%	18.66%
Grok 4.1 Fast	24.88%	18.66%	19.62%	14.35%	15.79%	11.00%
Claude Opus 4.5	8.61%	7.18%	7.66%	6.22%	6.70%	5.74%
Human (Average, pass@1)	68.96%	58.35%	-	-	-	-
Human ($\geq$ 2, pass@1)	93.5%	96%	-	-	-	-

Table 1: Experimental results on ARCANA. `CoT (language)` uses language-only chain-of-thought prompting; `CoT + Ana (VL)` adds an analogical prompt with visual inputs. `+Typo`/`-Typo` indicate whether minor typographical errors are counted as correct. `Total` is the union of tasks solved under the two setups (LLMs: **pass@12** in total across prompts, two settings, each with two prompt variants, each run three times; humans: **pass@1**).

5.2 Experiment analysis

The experimental results are shown in Table 1. Overall, LLM performance on ARCANA is below human performance in Table 1: the `pass@3` upper bounds of LLMs remain lower than the `pass@1` human (average) and human ($\geq$ 2). Nevertheless, the strongest models still solve a non-trivial subset of tasks, suggesting that current LLMs possess meaningful (albeit limited) capability for ARCANA-style problem solving. We emphasize that these results are influenced by evaluation budget (cost per task) and by the inherent stochasticity of LLM outputs.

Language vs. vision–language prompting. Generally, vision-language (VL) prompting yields lower performance than language-only prompting for most models (with one notable exception: GPT 5.2 improves under VL when typos are counted as correct). This trend is consistent with prior findings that adding visual inputs can reduce LLM performance Palmarini and Mitchell [2024], potentially due to misalignment between the vision and language modalities. Importantly, however, these aggregate comparisons do not imply that VL is strictly worse. The `Total` column is higher than either prompting alone, indicating that the two prompting modalities succeed on different subsets of tasks. Category-level inspection also suggests that some concepts benefit substantially from visual grounding (e.g., Gemini on `electric_clock` category and `moon_and_tide1` task, GPT on `pendulum` category). Together, these observations support our viewpoint that ARCANA is inherently visual in nature, yet not merely visual problems.

Correct analogies support LLMs solving ARCANA. Under analogical prompting, we collect and manually evaluate model-generated targets, and find that correct analogies tend to co-occur with correct answers, suggesting that analogy can materially support ARCANA task solving. The analogy accuracy could be find in Appendix C.5. We also observe substantial differences in analogical ability across models resulting in different result accuracy. Focusing on the two strongest models (GPT 5.2 and Gemini 3 Pro), the `pulley` category is particularly diagnostic: Gemini achieves 100% analogy correctness while GPT achieves 0%; correspondingly, Gemini reaches 67% answer accuracy in `pulley` whereas GPT remains at 0%. Overall, these results suggest that analogical reasoning is a meaningful bottleneck for ARCANA performance.

Category-wise performance and human-model gaps. We further examine performance by task category. Results on category-wise accuracy of humans versus LLMs can be found in Appendix C. Overall, human accuracy is relatively stable across categories, whereas LLM performance is more uneven. Humans outperform LLMs in most categories, but there are also categories in which some models match or exceed the human average (typically those dominated by precise, fine-grained manipulations, e.g., `shelter_then_move`). At the same time, several categories expose clear gaps where LLMs lag far behind humans (e.g., `water_deep_concept` and `animal`), indicating that substantial differences remain between current LLMs and human analogical reasoning.

More details are provided in the Appendix C. Nonetheless, the scale and breadth of modern LLMs suggest substantial potential for approaching human-like analogy. To narrow this performance gap and probe the limits of LLM analogical reasoning, we introduce a new reasoning framework, CONCEPTSIR, in the next section and study its empirical behavior.

6 CONCEPTSIR: A Reasoning Framework for Concept Slippage, Introspection, Reflection

Inspired by cognitive theories of analogy-making, especially Hofstadter and Mitchell's Copycat architecture Hofstadter et al. [2001, 1995], we propose CONCEPTSIR, an LLM-driven analogical reasoning framework built around Concept Slippage, Introspection, and Reflection. Following Copycat, where Workspace structures activate Slipnet concepts that may slip to related concepts and trigger Codelets, CONCEPTSIR treats the LLM's knowledge as an implicit Slipnet, context-conditioned generation as concept slippage, and concept execution as Codelets. Introspection and reflection then use execution feedback to iteratively refine conceptual hypotheses and executable rules. The framework is shown in Figure 4. Additional details are provided in Appendix D.

6.1 Concept Proposal and Slippage

Concept Slippage, a core mechanism in Copycat Hofstadter et al. [2001, 1995], describes how activating one concept can trigger related concepts and higher-level relational structures, thereby expanding the search over possible interpretations. This is particularly useful for LLMs, whose outputs can be overly context-dependent; increasing the diversity of candidate targets helps avoid getting trapped in a "local minima" of concepts.

Motivated by our human study and the breadth of LLM pretraining, we introduce a concept proposal and slippage stage to elicit diverse analogical targets. We first prompt candidates along four human-prior dimensions—color, shape, dynamics, and holistic recognition—and then use a slippage prompt to expand them into a broader concept neighborhood. From a Bayesian perspective, Copycat-style slippage can be viewed as combining conceptual priors with task evidence: the Slipnet encodes prior knowledge, while structures activated in the Workspace provide likelihood-like constraints on which concepts become relevant. Analogously, LLMs generate new concepts conditioned on both knowledge learned from pretraining and the given task context. Thus, CONCEPTSIR implements concept slippage as an additional prompting operation that leverages LLMs' context-conditioned generation to activate a more diverse set of related concepts.

6.2 Concept Execution

To reliably apply an inferred rule underlying all given demonstrations to the question grid, we represent the execution step as code rather than natural language. Natural-language rules are often ambiguous or internally inconsistent, making it difficult to ensure faithful, deterministic application

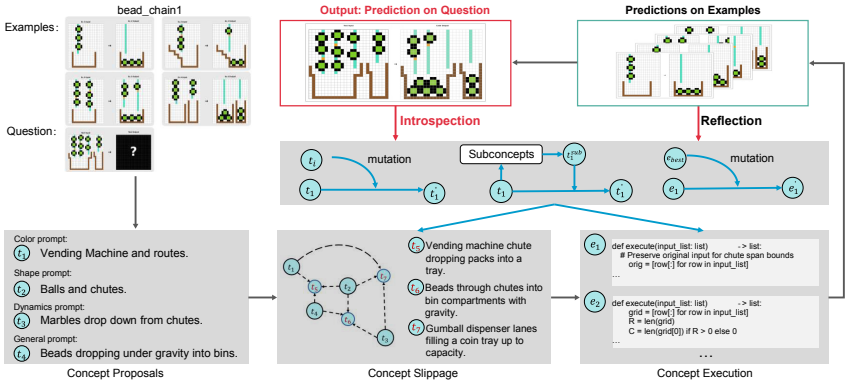


Figure 4: An overview of CONCEPTSIR framework. Given an ARCANA task, the framework first proposes initial concepts from multi-dimensional prompts. These concepts evolve via Concept Slippage to generate a diverse set of candidates, which are then implemented as executable code under Concept Execution. The system iteratively refines solutions through Reflection, employing mechanisms such as code mutation and concept mutation, while Introspection ensures the final prediction on the test question aligns with the established rules and human preference.

to grids. Using code provides an explicit, verifiable realization of a candidate rule and is broadly consistent with prior ARC solvers. Concretely, for each candidate target produced by the slippage stage, we prompt the LLM to synthesize a corresponding program as one possible instantiation of the target's rule, which is then executed and checked against the examples.

6.3 Introspection

Introspection iteratively aligns the model's analogical interpretation and its execution with human cognition (Principle 4). Many ARC tasks admit multiple rules consistent with the demonstrations, but humans typically favor one interpretation as the intended solution. For example, in a clock task, both "rotate the pointer on the right dial by $90°$" and "rotate the hour hand by $90°$" may fit all examples, yet the latter is more natural because "hour hand" composes more directly with the concept of a "clock". After each code update, we run the program on the examples and questions, using introspection to identify abnormal behaviors and producing targeted correction instructions; once all examples are matched, we perform a final introspection pass on the question output to filter invalid or human-misaligned solutions.

6.4 Reflection

Inspired by genetic algorithms Mitchell [1998], we perform two levels of probabilistic mutations in parallel: code mutation and concept mutation. At the code level, reflection repairs implementation and logic errors by comparing program outputs on the examples against ground truth, selecting the best-performing candidate as a reference (according to the prediction errors), and probabilistically mutating other candidates to explore alternative implementations. At the concept level, mutation includes broader slippage (to diversify targets) and finer-grained concept augmentation, which enriches a target with relevant sub-concepts needed for faithful execution (e.g., adding a "pile-up" concept on top of "gravity" for `bead_chain1`). Together, these mechanisms promote both diversity and implementability of candidate solutions.

6.5 Experimental results

We conduct a focused evaluation of CONCEPTSIR on a deliberately challenging subset of ARCANA in order to fully explore the capability of each modules. The subset consists of 17 tasks from 11 categories, many of which are unsolved or only rarely solved by direct LLM testing in Section 5. These tasks were randomly sampled with a preference for tasks with low direct-LLM accuracy. Under

a strict exact-match setting with one run per task and a budget of 5 loops, CONCEPTSIR solves $9/17$ tasks across 7 categories, requiring only 2.22 loops on average for successful cases. These results suggest that the structured cycle of concept proposal, slippage, execution, and refinement can help LLMs go beyond direct prompting, both enabling solutions on difficult tasks and improving proposal efficiency in several cases. Further analysis and details is provided in Appendix E.

Concept slippage contributes to program generation. We study tasks where failure is caused primarily by selecting an incorrect analogical target, e.g., `refraction2` and `bead_chain1`. In CONCEPTSIR, the concept proposal stage elicits multiple candidate targets, and concept slippage diversifies and refines them, enabling Gemini to recover the intended interpretations ("refraction" for `refraction2` and "beads falling into a container" for `bead_chain1`). Conditioned on these targets, GPT solves `refraction2` in two reflection steps and produces two correct programs for `bead_chain1` within three steps. In contrast, `refraction2` cannot be solved by all the LLM baselines, while `bead_chain1` is solved only once by GPT 5.2 across 12 attempts.

Introspection and reflection contribute to program modification. In our case studies, we find that introspection and reflection play a critical role in refining candidate programs into correct and generalizable solutions. For the `bead_chain1` task, the target concept of "beads falling into a container" is under-specified; reflection introduces an implicit discrete packing constraint (e.g., "bays behave like discrete parking slots"), which captures the pile-up mechanism required to implement the rule.

For `candle1`, none of the LLM baselines succeed, while human participants solve $5/6$ instances when typos are counted as correct. Even with the target concept provided, candidate programs can fit all training examples but fail on the test case, e.g., by deleting candle structures or producing inconsistent shadows. Introspection detects these failure modes and provides targeted correction instructions, while reflection revises the program without breaking consistency on the examples. In this case, reflection reveals that the candle should be "burned down" by a fixed number of cells, rather than simply shifting an orange segment downward.

7 Limitations

ARCANA is currently limited in scale and representation. Although it covers diverse targets, the space of real-world concepts and analogical sources is effectively unbounded, making larger benchmarks difficult without community effort. We view ARCANA as a starting point for such expansion. Its ARC-style grid abstraction also provides conciseness, precision, and cognitive challenge, but limits expressivity for structures such as oblique lines, circles, and ellipses, motivating richer representations for future far-analogy benchmarks.

We also note several limitations of CONCEPTSIR. Future work should evaluate it on a larger set of ARCANA tasks beyond the current challenging subset. Although code execution offers a faithful and reproducible way to apply hypothesized rules, it is not fully expressive for all ARCANA targets, especially topology-heavy ones; extending the executor to atomic grid operations with explicit where-to-apply instructions may improve generality. Moreover, our code-mutation procedure selects the "best" candidate solely by prediction error, which is a rigid proxy for solution quality. Additional components, such as explicit memory or human feedback, may further improve robustness and generalization.

8 Conclusion

We introduce ARCANA, a benchmark for evaluating far analogical reasoning, and analyze both human and LLM performance. We further propose CONCEPTSIR, an LLM-driven framework based on concept slippage, introspection, and reflection, showing that it can solve tasks missed by direct LLM baselines. We hope ARCANA and CONCEPTSIR help establish far analogy as an important frontier for more human-like, general reasoning.

Acknowledgments and Disclosure of Funding

This work was partially supported by the National Science Foundation of China project (32441106) and the National Science and Technology Major Project (2022ZD0114900). The authors thank Dr. Yi-Long Lu at BIGAI for the discussion on the experiment design.

References

Dedre Gentner. Structure-mapping: A theoretical framework for analogy. Cognitive Science, 7(2): 155–170, 1983. ISSN 0364-0213. doi: https://doi.org/10.1016/S0364-0213(83)80009-3. URL https://www.sciencedirect.com/science/article/pii/S0364021383800093.

Douglas R Hofstadter et al. Analogy as the core of cognition. The analogical mind: Perspectives from cognitive science, pages 499–538, 2001.

Douglas R Hofstadter, Melanie Mitchell, et al. The copycat project: A model of mental fluidity and analogy-making. Advances in connectionist and neural computation theory, 2(205-267):2–3, 1995.

Keith J Holyoak and Paul Thagard. Analogical mapping by constraint satisfaction. Cognitive science, 13(3):295–355, 1989.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity's last exam. arXiv preprint arXiv:2501.14249, 2025.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? arXiv preprint arXiv:2509.16941, 2025.

Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-agi-2: A new challenge for frontier ai reasoning systems. arXiv preprint arXiv:2505.11831, 2025.

Kenneth D Forbus, Ronald W Ferguson, Andrew Lovett, and Dedre Gentner. Extending sme to handle large-scale cognitive modeling. Cognitive Science, 41(5):1152–1201, 2017.

John and Jean Raven. Raven progressive matrices. In Handbook of nonverbal assessment, pages 223–237. Springer, 2003.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5317–5327, 2019.

François Chollet. On the measure of intelligence. arXiv preprint arXiv:1911.01547, 2019.

Keya Hu, Ali Cy, Linlu Qiu, Xiaoman Delores Ding, Runqian Wang, Yeyin Eva Zhu, Jacob Andreas, and Kaiming He. Arc is a vision problem! arXiv preprint arXiv:2511.14761, 2025.

Isaac Liao and Albert Gu. Arc-agi without pretraining. arXiv preprint arXiv:2512.06104, 2025.

Hongjing Lu, Ying Nian Wu, and Keith J Holyoak. Emergence of analogy from relation learning. Proceedings of the National Academy of Sciences, 116(10):4176–4181, 2019.

Maxwell Crouse, Constantine Nakos, Ibrahim Abdelaziz, and Ken Forbus. Neural analogical matching. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 809–817, 2021.

Alessandro B Palmarini and Melanie Mitchell. Abstract understanding of core-knowledge concepts: Humans vs. llms. In ICML 2024 Workshop on LLMs and Cognition, 2024.

Melanie Mitchell. An introduction to genetic algorithms. MIT press, 1998.

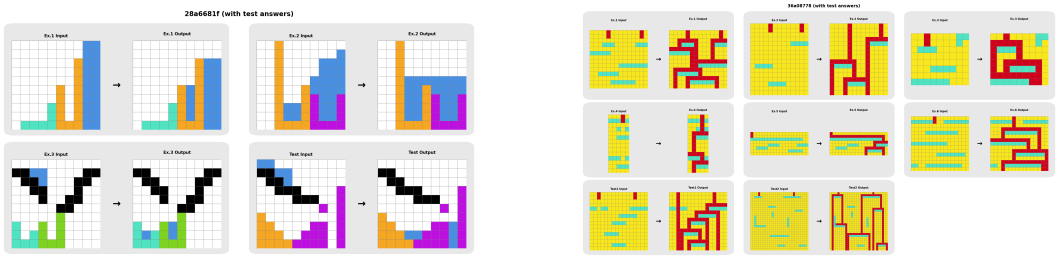
Douglas Hofstadter and Emmanuel Sander. Surfaces and essences: Analogy as the fuel and fire of thinking. Basic Books (AZ), 2013.

Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H Chi, and Denny Zhou. Large language models as analogical reasoners. arXiv preprint arXiv:2310.01714, 2023.

A Motivation of ARCANA

The grid format is following Chollet's motivation Chollet [2019]: grids provide a concise, precise, and cognitively challenging format for evaluating abstraction and reasoning, as we stated in Section 7. More importantly, as Hofstadter and Sander argue in Surfaces and Essences Hofstadter and Sander [2013], abstraction is deeply grounded in analogy. We therefore use the grid as an abstract target domain to which relational structures from real-world concepts can be transferred.

We have consistently acknowledged ARC-AGI as an important inspiration for ARCANA. In particular, ARC-AGI-2 task 28a6618f, which admits a natural water-related interpretation, directly inspired our design; among the public ARC-AGI-1&2 tasks, we found only one other similarly explicit case, 36a08778.



(a) 28a6681f (b) 36a08778

Figure 5: Visualization of two ARC-AGI-2 task instances that may admit explicit far-analogy interpretations. Both tasks can be understood through analogies to water: (a) water flow and (b) a water stream.

B Human study details

B.1 Quantification details.

Excluding typo responses, we compute the Pearson coefficient ϕ for a 2×2 contingency table of answer correctness versus analogy correctness:

$$\phi = \frac{P_{CC} - P_{CA}P_{AC}}{\sqrt{P_{CA}(1 - P_{CA})P_{AC}(1 - P_{AC})}}, \tag{1}$$

where CA denotes all correct answers, AC denotes all analogy-correct answers, and P denotes proportions in the filtered set. In our data, $\phi = 0.391$. A chi-square test derived from $\chi^2(1) = n\phi^2$ yields $\chi^2(1) \approx 226.9$ with $p \ll 0.001$. We also report an odds ratio of $OR \approx 25.6$ (95% CI $\approx [14.3, 45.8]$) and Yule's $Q \approx 0.93$.

B.2 Experimental details.

Participants were recruited through an official university platform and completed the study on a custom-built website. A screenshot of the task interface of human study is shown in Figure 6. Participants are asked to infer the rule from input–output examples, constructed the test grid, and described what the task reminded them of in the real world. The grid editor and analogy field appeared on the same page, with no enforced response order. Participants could save progress and had 120 hours to complete the study. The website source code has been released in github.

Informed Consent for Participants.

Please read the following information carefully before deciding whether to participate. Your participation is voluntary, and you may stop or withdraw at any time without penalty. Note that eligibility for the study payment is described below.

Purpose This study aims to collect human responses on a set of short reasoning tasks. The data will be used to analyze human problem-solving behavior and to support research in cognitive science and related fields.

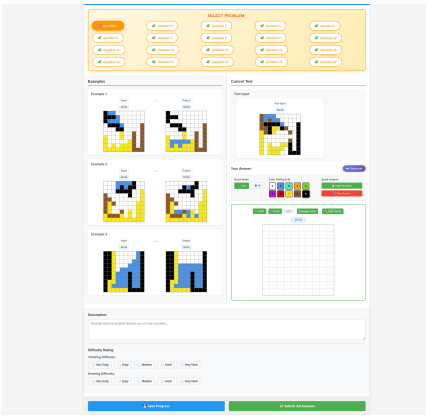


Figure 6: A screenshot of the formal task interface.

Procedures If you agree to participate, the testing platform will guide you through the following sequence of pages and steps:

Introduction page: An introduction page describes the study and tips for participation.

Tutorial page: A short interactive tutorial that demonstrates how to complete the tasks. You must read and follow the tutorial carefully to learn the response format and controls.

Attention check page: A short verification page to confirm that you understood the instructions in the tutorial. You must complete the attention check correctly to continue. Formal test page: If you pass the attention check, you will be assigned the formal test sequence of tasks.

Important: You must read the tutorial and complete the attention check correctly to proceed to the formal test. Participants who do not complete the tutorial or who fail the attention check will not proceed to the formal test and will not be eligible for compensation.

Data Collected Task responses (the answers you provide for each task).

Timestamps for task start and completion.

Non-identifying metadata such as browser type and assignment ID. These identifiers are used for assignment management and quality control only and are not direct personal identifiers.

Confidentiality All data will be treated as confidential. We will only collect necessary data for data management. These will not be used to direct personal identification in research outputs and will be handled in accordance with the project's data-management policies. Collected research data will be stored on secure drives and used only for approved research purposes. Access to raw data will be restricted to authorized research personnel. Any publications or publicly shared datasets will contain only aggregated or anonymized results.

Compensation All participants who successfully complete and submit the assigned test session will receive a fixed base payment for their time. In addition to this base payment, participants will earn a small bonus for each task answered correctly. Your total payment will be calculated as the base payment for the completed submission plus the per-item bonuses for correctly answered tasks.

The exact base amount and the bonus rate per correct answer will be displayed on the testing platform and in any payment instructions before you begin. To be eligible for payment (including any earned bonuses), you must complete the assigned session and submit your responses according to the study instructions; withdrawing before completion or failing to submit may affect eligibility. Payments will be issued by the method described by the study coordinator and may be subject to verification and routine quality-control checks.

Risks and Benefits There are minimal risks associated with this study. You may experience mild fatigue from performing tasks. There is no direct benefit to you, but your participation contributes to scientific understanding of reasoning.

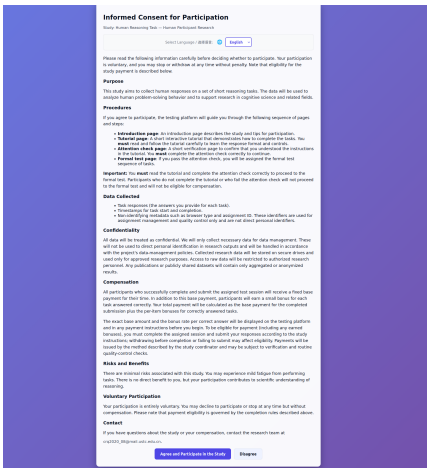


Figure 7: A screenshot of informed consent for participants.

Voluntary Participation Your participation is entirely voluntary. You may decline to participate or stop at any time but without compensation. Please note that payment eligibility is governed by the completion rules described above.

Instructions and Tips

Instructions:

1. You will be presented with visual pattern recognition problems
2. Each problem contains example input-output pairs to help you understand the pattern
3. Your task is to identify the underlying rule and apply it to new test cases
4. Use the color palette and drawing tools to create your answers
5. After solving all problems, describe what each problem reminds you of in the real world
6. Once you submit your answers, you cannot return to edit them
7. You can save your progress at any time and return later, but please finish the formal test within 120 hours.

Tips:

1. Study the examples carefully to understand the transformation pattern.
2. Look for spatial relationships, color changes, and geometric patterns.
3. Click on any example to enlarge it - easier to see details in small grids.
4. Take your time - you have 120 hours to complete the test. If you experience mild fatigue, take a short break and return later.
5. More compensation for more correct answers!

Tutorials

Welcome to Tutorial! Step 1 of 6 Welcome to the Cognitive Reasoning Tutorial! This interactive guide will walk you through all interface features.

Example Area: Here you'll see example input-output pairs that help you understand the pattern. Click any example to enlarge it for better visibility, and drag the grid around to explore different parts between input and output of the example.

Problem Selection Step 2 of 6 Use these tabs to switch between different problems. Each problem has its own examples and tests.

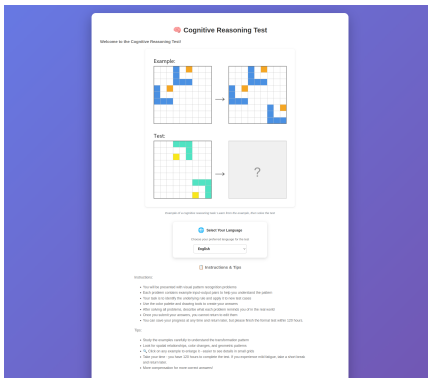


Figure 8: A screenshot of instructions page.

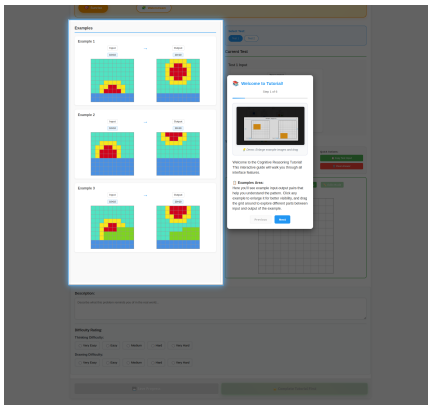


Figure 9: A screenshot of tutorial.

Test and Test Navigation Step 3 of 6 This area shows the current test case - the input grid you need to transform. Study it carefully along with the examples.

Click to Enlarge: Click on the test input to open a draggable floating window with an enlarged view. You can move this window around and keep it open wile working on your answer!

Sometimes a problem has multiple tests, use the tabs above to switch between them. Each test requires a separate solution. Watch how the "Current Test" area updates when you switch tests!

Drawing Tools Step 4 of 6 Use these tools to create your answer:

1. Paint/Fill mode buttons
2. Color palette
3. Copy and Clear buttons
4. Enlarge View button
5. **Edit Mode button** (unlocked after tutorial)

Tip: To get started quickly, click the **Copy Test Input** button to copy the test input into the answer grid as a starting point. When you have finished your answer, use the **Enlarge View** button to open an enlarged side-by-side comparison of the test input and your answer so you can check accuracy.

Here are also some keyboard shortcuts for efficient solving:

Color Selection:

1. `0-9` Select color directly
2. `W/A/S/D` Navigation on the color palette

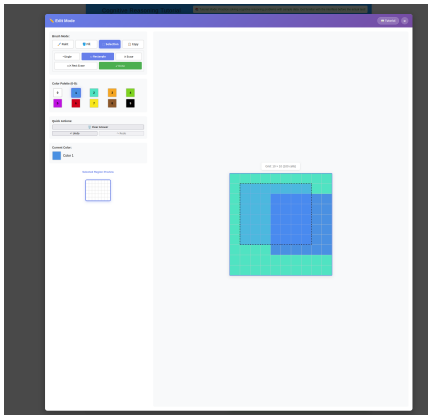


Figure 10: A screenshot of edit mode.

Drawing Tools:

1. `Q` Paint mode (single cell)
2. `E` Fill mode (flood fill)

Edit History:

1. `Ctrl+Z` Undo last action
2. `Ctrl+Y/Ctrl+Shift+Z` Redo action
3. `Ctrl+Shift+X` Copy current problem input

Now enjoy your drawing! *To explore more features, please access Edit Mode after completing the tutorial.*

Description Field and Difficulty Rating Step 5 of 6 After solving each problem, please describe what the problem reminds you of in the real world. This helps researchers understand how people interpret these patterns and will not affect your accuracy.

Additionally, please rate the **Thinking Difficulty** and **Drawing Difficulty** of each problem on a 5-point scale from 'Very Easy' to "Very Hard". These ratings are required and help us understand your perception of the difficulty distribution of the problems.

Tutorial Complete! Step 6 of 6 Congratulations! You've mastered all the interface features:

1. **Examples:** Viewing and enlarging example patterns
2. **Navigation:** Switching between problems and tests
3. **Drawing Tools:** Paint/Fill modes and color palette
4. **Description:** Writing pattern observations

Important: Save Your Progress!

1. Remember to **save** before closing the page (use the Save Progress button or Ctrl+S). When you reopen the page, your saved progress will automatically loaded
2. There is NO auto-save, you must manually save your work. Closing the page without saving will result in **losing** your unsaved progress.

Ready to begin? Click "Begin Attention Check" to start with the attention verification before the main test!

Edit Mode Tutorial

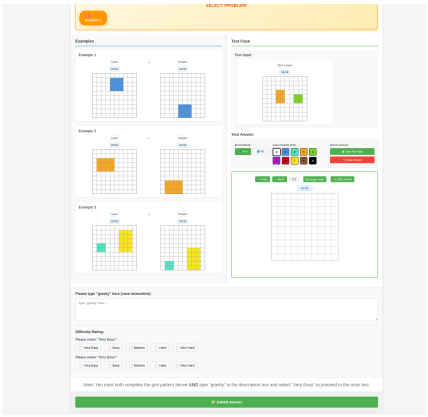


Figure 11: A screenshot of attention check.

Welcome to Edit Mode! Step 1 of 8 Welcome to Edit Mode! This focused editing mode retains most of the features and shortcuts while adding powerful new **selection and copy** capabilities. Let's explore these new features to make your grid editing more efficient!

Single Cell Selection Step 2 of 8 When in selection mode, you can use **Single** selection to choose any shape of region. Click each cell to add it to your selection one by one, then click the **Done** button when finished.

A thumbnail of your selected region will appear below the toolbar. Click the thumbnail to view an enlarged window for easy inspection of your selection.

Rectangle Selection Step 3 of 8 You can also use **Rectangle** selectionmode to quickly select a rectangular region.

1. **Left-click** to set the starting corner
2. **Left-click** again to set the ending corner (diagonal)
3. **Right-click** to cancel the current selection

Erasing Selections Step 4 of 8 If you want to remove parts of your selection, you can use **Erase** mode to delete individual cells, or use **Rectangle Erase** mode to quickly remove rectangular regions.

Useful Shortcuts:

1. `A/D` Switch between selection submodes
2. `F` Finish/confirm selection

Copy & Paste Step 5 of 8 When you have a selected region, you can choose **Copy** mode to apply it.

In copy mode, your selected region appears on the grid. You can **drag your mouses** to any position on the grid to move the selection. Once you find the perfect spot, click **Confirm Copy** to complete the operation.

Switch to any other mode to see the copied result!

Mirror Transforms Step 6 of 8 You can use **Horizontal Mirror** and **Vertical Mirror** to quickly flip your selection and create symmetrical patterns!

This is especially useful for creating symmetric designs and saves tons of time on repetitive operations.

Useful Shortcuts:

1. `W` Vertical mirror (flip up-down)
2. `S` Horizontal mirror (flip left-right)

Rotation Transforms Step 7 of 8 You can also use **Rotate Clockwise** and **Rotate Counter-Clockwise** to rotate your selection by 90 degrees.

After rotation, the selection automatically adjusts its position to stay within grid boundaries - smart and convenient!

Useful Shortcuts:

1. `A` Rotate counter-clockwise 90 degrees
2. `D` Rotate clockwise 90 degrees

Tutorial Complete! Step 8 of 8 Congratulations on completing the Edit Mode tutorial! You've now mastered:

1. **Selection Tools** Single/Rectangular selection & erase
2. **Copy & Paste** Flexible region copying
3. **Mirror Transforms** Horizontal/Vertical flips
4. **Rotation Transforms** Clockwise/Counter-clockwise rotation

Use these powerful new features with their shortcuts to edit grids more efficiently and create more complex patterns!

Tip: Press `Q` or `E` to quickly switch between these modes!

Attention check

The participants are required to finish a task concerned with gravity, and correctly type the "gravity" in the description frame. For difficulty rating, the participants are required to choose "Very Easy" for both thinking difficulty and drawing difficulty.

All these text are provided in both English and Chinese, and you may find it out in the source code.

B.3 Analysis details.

All responses are evaluated by one author using consistent criteria, and we acknowledge the possibility of subjective annotation bias. Inter-rater agreement could not be measured. An analogy description is marked correct if it captures the core transformation rule; responses that do not express the core rule, including "I do not know," are marked incorrect. Ambiguous cases and essential errors are conservatively classified as incorrect, and we also report strict exact-match accuracy without typo handling.

Overall, humans achieve 68.96% average accuracy on ARCANA, including 10.61% typo-level matches, while the Human ($\geq$ 2 humans) accuracy reaches 96%, including 2.5% typo-level matches. The results on Human ($\geq$ 2 humans) show that typo judgments do not drive the conclusions of ARCANA's validity. In Figure 3(a), each histogram bin aggregates tasks by their per-task accuracy (binned in 10% intervals). For subjective difficulty, participants separately rate thinking and drawing difficulty using a five-level scale (very easy, easy, medium, hard, very hard), which we map to scores 1-5. For each task, we average the two scores to obtain a single difficulty value. Figure 3(b) plots each task as one point and overlays fitted trend lines under the two Typo conventions.

B.4 ARCANA tasks have only one and clear ground truth.

In ARCANA, the ground truth is designed with the deterministic grid-level transformation rule for each task. In our human study, we provided participants with the ground-truth solutions after testing, and we received no feedback suggesting that the answers were ambiguous. When some instances feel "unclear" to participants during testing , we believe this is largely driven by variability in far-analogy mapping (i.e., differences in concept familiarity and in identifying the intended target concept) rather than ambiguity in the grid-level rule itself. The observed accuracy primarily reflects participants' capabilities.

In alignment with the construction principles of ARCANA, we will explain the ARCANA evaluation methods in detail.

"Multiple plausible analogical targets" do not imply multiple valid grid outputs. In the paper we mainly refer to the "multiple plausible analogical targets" but not "solutions". We mean that different participants may describe the same task using different analogical targets, while still implying the same underlying grid-level transformation rule (Construction Principle 3: multiple targets, one rule). For example, in cat1, some participants described the patterns as "cats", while others described them as "little devils". These different targets can still yield the same correct grid transformation, and we treat both target descriptions as plausible as long as they faithfully reflect the intended rule.

For the grid answers, ARCANA is designed to have a unique ground-truth output under strict evaluation. For every task, we aim to ensure a single intended grid-level solution. Any designer-perceived "alternative" rules that might appear plausible are excluded by adding or modifying examples that serve as counter-evidence, so that only one rule remains consistent with the examples. While a single designer cannot guarantee absolute completeness, we made a strong effort to consider plausible alternatives and add counter-evidence examples where needed. Additionally, after the human study we provided participants with the ground-truth solutions, and we received no feedback indicating that any answers were ambiguous.

When multiple rules fit the examples, we select the human-aligned one—this is different from accepting multiple outputs. The issue of "multiple rules consistent with the examples" (Construction Principle 4: Human-aligned correctness) is closer to the cases where multiple rules might fit all the examples. In such cases, we resolve the under-specification during dataset construction by selecting the rule that is most natural to humans and best aligned with everyday concepts (i.e., lowest in cognitive burden), rather than treating multiple grid outputs as valid at evaluation time. For instance, in clock1 task, both "rotate the pointer on the right dial by $90°$" and "rotate the hour hand by $90°$" may fit all examples, yet we select the latter as more natural and concept-aligned (as noted in the appendix). This motivation also informs why ConceptSIR includes an introspection step.

We do NOT accept unexpected grid solutions as correct. Accordingly, grid answers are scored by strict exact-match against the single ground-truth output. If we ever observe a non-matching output that seems plausibly valid, we treat it as evidence that the task is under-specified and would revise the task by adding/modifying examples and re-testing, rather than accepting multiple outputs at evaluation time. In contrast, for analogy targets, multiple descriptions are acceptable as long as they correctly reflect the task rule.

Thus, human judgment is used only for **typo annotation** (and ambiguous cases are counted as incorrect), not for validating alternative grid-level solutions nor as a "trick" to make the accuracy results seem better. If multiple outputs appear valid, we treat it as a dataset issue to be fixed rather than an evaluation-time exception.

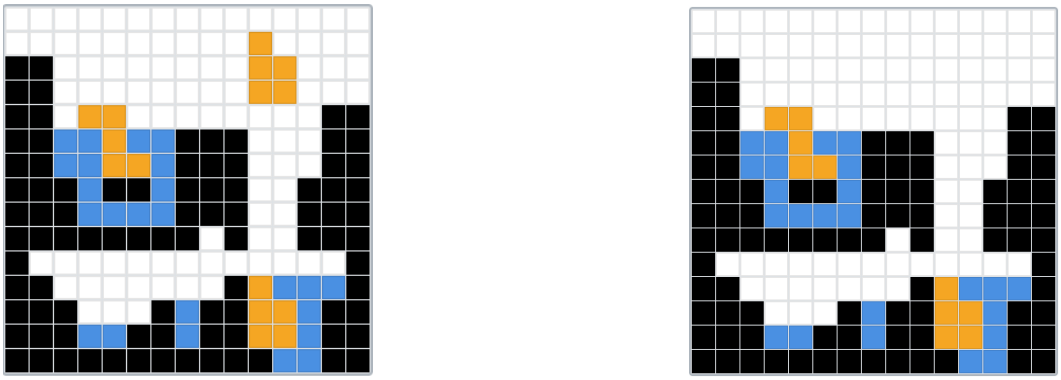
B.5 Typo taxonomy

To make our use of "typo" more interpretable, we provide a rough taxonomy and concrete examples below (task names in parentheses). We believe that explicitly enumerating recurring patterns makes typo handling more transparent. Futhermore, we provide some typos in Figure 12, Figure 13, Figure 14, Figure 15, Figure 16.

1. **Missed erase / incomplete edit after a state change.**
   (a) Object falls into water but the pre-fall object is not erased (archimedes1).
   (b) Dye droplet falls into water but the falling droplet is not erased (dye1).
   (c) Magnet gets attracted but the previous magnet is not erased (magnetic_or_chemical_reaction1).
2. **Accidental clicks or small omissions in complex structures.**
   (a) A bottle-wall cell is painted as “empty” blue without affecting the pouring result (bottle3).
   (b) One source cell of water is missing when drawing the water flow (bottle3).
3. **Color slip that flips a single cell to an adjacent/nearby color.**
   (a) When filling water inside a container, a floor cell is accidentally colored as water (water_stream1).

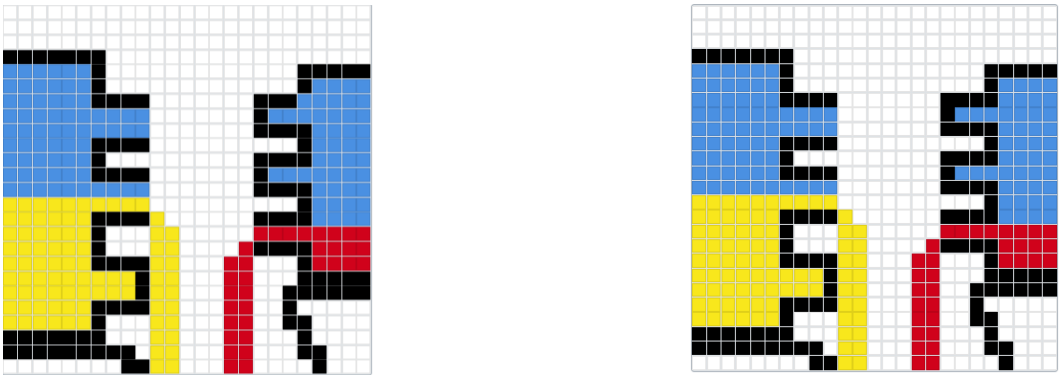
(b) During a rotation, the rotation-center cell changes from rope color to wall color (ballistic_pendulum1).

4. **Off-by-one / off-by-two after translation or rotation of a pattern.**
   (a) After translating the fish pattern, the tail is off by one cell (fish1).
   (b) After wheel rotation, an extra wheel-center cell appears (gear2).
5. **Failure to apply a copy operation, causing minor irrelevant errors.**
   (a) Cups are stacked correctly, but a one-cell-high floor is not drawn (cup2).
   (b) One or two single cells in complicated pattern transformation (kaleidoscope1).



(a) Typo answer. (b) Correct answer.

Figure 12: A typo of **missed erase / incomplete edit after a state change** in archimedes1.



(a) Typo answer. (b) Correct answer.

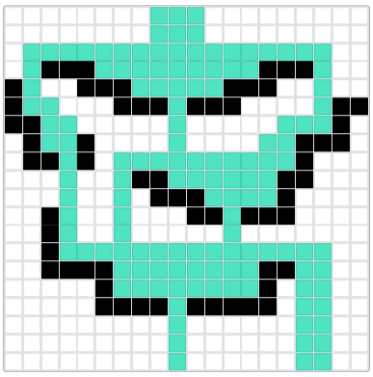
Figure 13: A typo of **Accidental clicks or small omissions in complex structures** in bottle3.

For ambiguous cases where it is difficult to determine whether an error should be considered a typo, we conservatively treat the answer as incorrect. Moreover, we always report strict exact-match accuracy without typo handling.

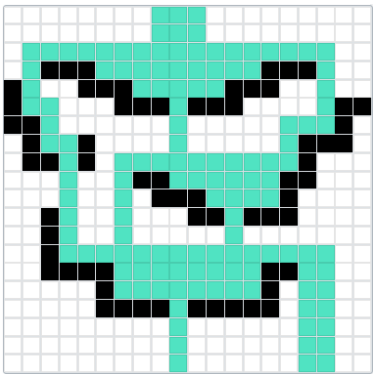
C LLMs evaluation details

C.1 Cost and tokens that LLMs spend.

We provide the cost and tokens of every LLMs and human in Table 2 for comparison. The currency in the table is USD. All tests are finished through OpenRouter.

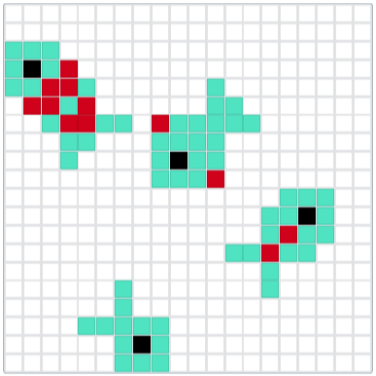


(a) Typo answer.

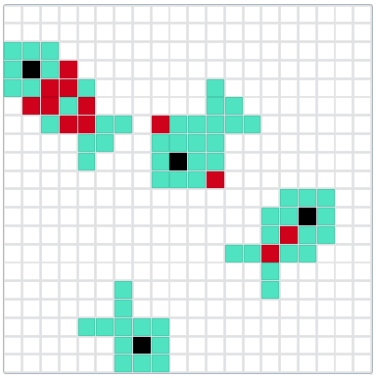


(b) Correct answer.

Figure 14: A typo of **Color slip that flips a single cell to an adjacent/nearby color** in water_stream1.

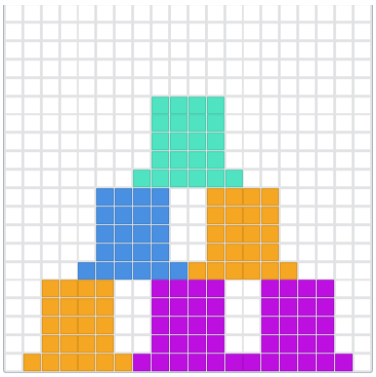


(a) Typo answer.

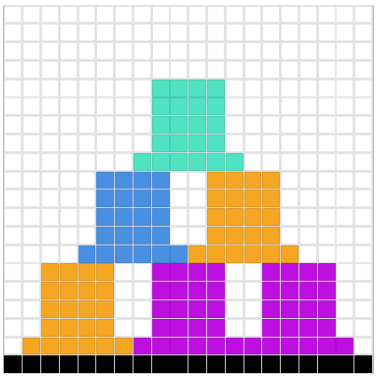


(b) Correct answer.

Figure 15: A typo of **Off-by-one / off-by-two after translation or rotation of a pattern** in fish1.



(a) Typo answer.



(b) Correct answer.

Figure 16: A typo of **Failure to apply a copy operation, causing minor irrelevant errors** in cup2.

	GPT 5.2	Gemini 3.1 pro	Grok 4.1 Fast	Claude Opus 4.5	human
cost/task	**11.84**	4.33	1.16	1.40	6.93
tokens/task	571K	475K	**2.46M**	169K	-

Table 2: The average cost and tokens LLMs and human spend per task.

C.2 LLMs results and baseline prompts

While chain-of-thought prompts might be easy to understand, analogical prompts actually include the prompt about associating the task with real-world objects or phenomena which is targets. Beyond the two high-level prompting setups reported in Table 1—`CoT (language)` and `CoT, ana (VL)`—we additionally use two prompt variants within each setup. For each variant, we compute `pass@3` and obtain a set of tasks solved by that variant. The reported score for a setup is then computed from the union of these task-level success sets (i.e., a task is counted as solved if it is solved by either prompt variant under `pass@3`). We also observe occasional "truncated" outputs (the matrix is cut off mid-generation) and formatting errors (e.g., mismatched brackets). Cost and token statistics include all attempts, including failures.

Because our evaluation uses a fixed number of trials (`pass@3`) under specific prompting setups, the reported accuracies should be viewed as an empirical upper bound under our testing budget rather than an intrinsic ceiling of each model. Differences in cost per task can also affect how aggressively one can sample, refine, or verify outputs, further entangling performance with budget. In addition, run-to-run variability is substantial; therefore, the analysis that follows is intended to surface plausible tendencies and failure modes observed in our runs, not to claim universal properties of the models.

The prompts used in baseline are provided below:

CoT system prompt 1:

> You are an expert ARC (Abstraction and Reasoning Corpus) task solver. I will provide you with the matrix of the question, you could decide the rule to solve the question. In matrix, every example is provided as input-output pair, and question is provided as input. There might be multiple questions, they all share the same rule as the examples. Please follow the principles and answer the given test.
> Principles:
> - Avoid ad-hoc exceptions that only fix one example but break generality.
> - If there are multiple tests, your answer could look like: [grid1, grid2]
> Output format:
> ANSWER:
> [Your answer for all tests - no explanations and no markdown formatting. If there are multiple tests, follow the principles to sequentially place the answers, and your answer should be like [grid1, grid2] where grid1 and grid2 are the answers of different questions.]

CoT system prompt 2:

> You are an expert ARC (Abstraction and Reasoning Corpus) task solver. I will provide you with the matrix of the question, you could decide the rule to solve the question. In matrix, every example is provided as input-output pair, and question is provided as input. There might be multiple questions, they all share the same rule as the examples.
> Your mandatory workflow:
> 1. Understand:
> - Carefully inspect all training input-output pairs.
> - Describe in works what changes from input to output.
> 2. Hypothesize:
> - Propose at least two distinct candidate rules that would explain the transformations.
> - For each candidate, give a clear natural language and short pseudocode.
> - Guarantee the diversity of the different candidate rules, try to come up with more rules from diverse aspects.
> 3. Test:
> - For each candidate rule, mentally apply it to every training input.
> - Explicitly compare the predicted output with the ground-truth output.
> - Mark each candidate as 'fully consistent', 'partially consistent', or 'inconsistent'.
> 4. Select:
> - Choose the simplest rule that is fully consistent with all training examples.
> -If none is fully consistent, refine or combine candidates until at least one fits all data.

5.Apply:
- Apply the selected rule to each test input grid to obtain predicted test outputs.
Principles:
- Never stop at the first plausible rule; always consider multiple hypotheses.
- Avoid ad-hoc exceptions that only fix one example but break generality.
- If there are multiple tests, your answer could look like: [grid1, grid2]
- Prefer object-based reasoning (connected components, shapes, repetition, symmetry, etc.)
Output format:
REASONING:
[Your reasoning progress here, include all of your workflow.]
ANSWER:
[Your answer for all tests - no explanations and no markdown formatting. If there are multiple tests, follow the principles to sequentially place the answers, and your answer should be like [grid1, grid2] where grid1 and grid2 are the answers of different questions.]

CoT, Analogy system prompt 1:

You are an expert ARC (Abstraction and Reasoning Corpus) task solver. Now you will be provided with one ARC task, the rule of the task could be analogous to a target in real life which could be some objects or some natural phenomena. I will provide you with the matrix of the question, you could decide the rule to solve the question. In matrix, every example is provided as input-output pair, and question is provided as input. There might be multiple questions, they all share the same rule as the examples. Please follow the principles and answer the given test. You are encouraged to use analogy to complete the test.
You will be provided with the visualization of the examples and the questions for better analogy. In the visualization of examples, every column is an input-output pair, the first row is input and the second row is output. From left to right, the columns correspond to example1, example2, and so on. In visualization of questions, there is only one row for the reason that there is no ground truth. Also, from left to right, the columns correspond to question1, question2, and so on.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Principles:
- Avoid ad-hoc exceptions that only fix one example but break generality.
- If there are multiple tests, your answer could look like: [grid1, grid2]
Output format:
ANALOGY:
[Your analogous target here, use one word or sentence to describe the rule.]
ANSWER:
[Your answer for all tests - no explanations and no markdown formatting. If there are multiple tests, follow the principles to sequentially place the answers, and your answer should be like [grid1, grid2] where grid1 and grid2 are the answers of different questions.]

CoT, Analogy system Prompt 2:

You are an expert ARC (Abstraction and Reasoning Corpus) task solver. Now you will be provided with one ARC task, the rule of the task could be analogous to a target in real life which could be some objects or some natural phenomena. I will provide you with the matrix of the question, you could decide the rule to solve the question. In matrix, every example is provided as input-output pair, and question is provided as input. There might be multiple questions, they all share the same rule as the examples. Please follow the principles and answer the given test.

Your mandatory workflow:
1. Understand:
- Carefully inspect all training input-output pairs.
- Describe in works what changes from input to output.
2. Hypothesize:
- Propose at least four distinct candidate analogies and their corresponding rules that would explain the transformations.
- For each candidate, give a clear natural language and short pseudocode.
- Guarantee the diversity of the different candidate rules, try to come up with more rules from diverse aspects.
3. Test:
- For each candidate rule, mentally apply it to every training input.
- Explicitly compare the predicted output with the ground-truth output.
- Mark each candidate as 'fully consistent', 'partially consistent', or 'inconsistent'.
4. Select:
- Choose the simplest rule that is fully consistent with all training examples.
-If none is fully consistent, refine or combine candidates until at least one fits all data.
5.Apply:
- Apply the selected rule to each test input grid to obtain predicted test outputs.
You will be provided with the visualization of the examples and the questions for better analogy. In the visualization of examples, every column is an input-output pair, the first row is input and the second row is output. From left to right, the columns correspond to example1, example2, and so on. In visualization of questions, there is only one row for the reason that there is no ground truth. Also, from left to right, the columns correspond to question1, question2, and so on.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Principles:
- Never stop at the first plausible rule; always consider multiple hypotheses.
- Avoid ad-hoc exceptions that only fix one example but break generality.
- If there are multiple tests, your answer could look like: [grid1, grid2]
- Prefer object-based reasoning (connected components, shapes, repetition, symmetry, etc.)
Output format:
REASONING:
[Your reasoning progress here, include all of your workflow.]
ANALOGY:
[Your analogous target here, use one word or sentence to describe the rule.]
ANSWER:
[Your answer for all tests - no explanations and no markdown formatting. If there are multiple tests, follow the principles to sequentially place the answers, and your answer should be like [grid1, grid2] where grid1 and grid2 are the answers of different questions.]

C.3 Response-level accuracy

Noted that the LLM accuracy and human accuracy are under different metrics in Table 7 (pass@12 vs. pass@1), we report the response-level accuracy across all outputs of the LLMs. We believe this will provide an empirical estimate of single-response accuracy across repeated runs.

C.4 Differences between language and vision-language prompting

To localize modality effects, we further inspect which task categories are solved under each prompting modality for Gemini and GPT. While most categories exhibit only small differences, we identify several cases with large modality gaps. In the `electric_clock` category, Gemini achieves 0% accuracy under language-only prompting but 100% under vision–language prompting. Similarly, in the `pendulum` category, GPT improves from 0% under language-only prompting to 75% under vision–language prompting. For `moon_and_tide1`, Gemini attains 100% under vision–language prompting but fails under language-only prompting, and its successful response explicitly attributes the pattern to lunar-phase variation.

Model	ARCANA					
	Total		CoT (language)		CoT + Ana (VL)	
	+Typo	-Typo	+Typo	-Typo	+Typo	-Typo
GPT 5.2	**18.86%**	**17.03%**	**17.54%**	**15.87%**	**20.18%**	**18.18%**
Gemini 3 Pro	13.12%	8.57%	13.96%	9.17%	12.28%	7.97%
Grok 4.1 Fast	6.46%	4.39%	6.70%	4.47%	6.22%	4.31%
Claude Opus 4.5	2.23%	1.71%	2.47%	1.99%	1.99%	1.44%
Human (Average, pass@1)	68.96%	58.35%	-	-	-	-

Table 3: Response-level accuracy on ARCANA. For each LLM, Total aggregates all 12 responses per task across four prompt variants and three runs. `CoT (language)` uses language-only chain-of-thought prompting; `CoT + Ana (VL)` adds an analogical prompt with visual inputs. Missing or failed generations are counted as incorrect. `+Typo` counts minor typographical errors as correct, whereas `-Typo` uses strict exact match.

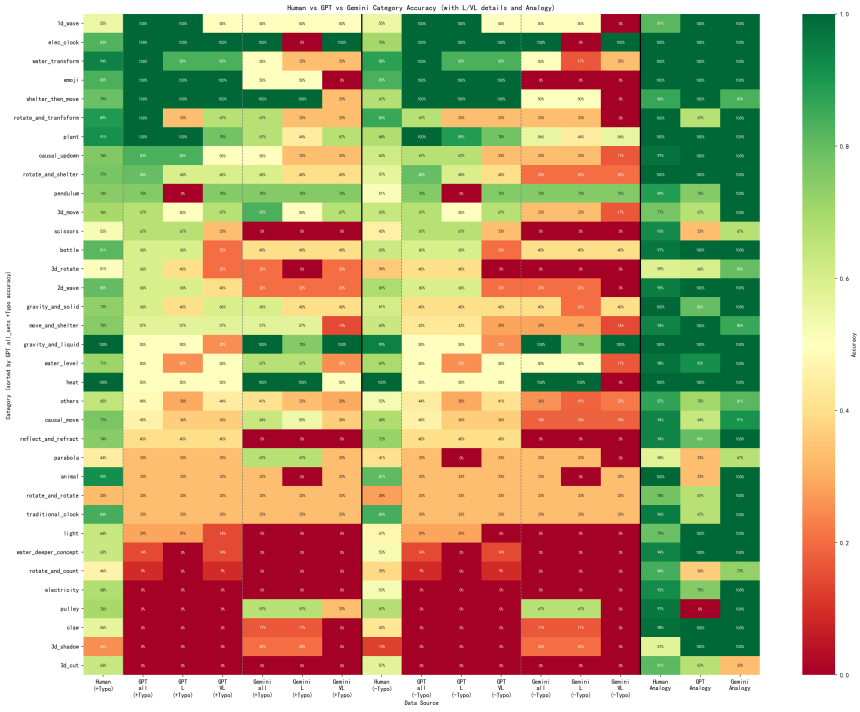


Figure 17: Accuracy distribution of GPT 5.2, Gemini 3 Pro and human across various task categories, including answers accuracy and analogy accuracy.

Conversely, we also observe categories where performance decreases after introducing vision-language prompting. These decreases do not follow a consistent pattern across Gemini and GPT, suggesting that the impact of visual inputs is model-dependent and intertwined with output stochasticity, rather than reflecting a uniform modality effect.

C.5 Detailed analysis on analogy

All results are provided in Figure 17, including the accuracy of analogy and answers on different categories for human, GPT 5.2, and Gemini 3 Pro. We compare GPT 5.2 and Gemini 3 Pro analogy accuracy across task categories using a scatter plot (Figure 18), where each point corresponds to a category. Points above the diagonal indicate categories in which Gemini's analogy accuracy exceeds GPT's.

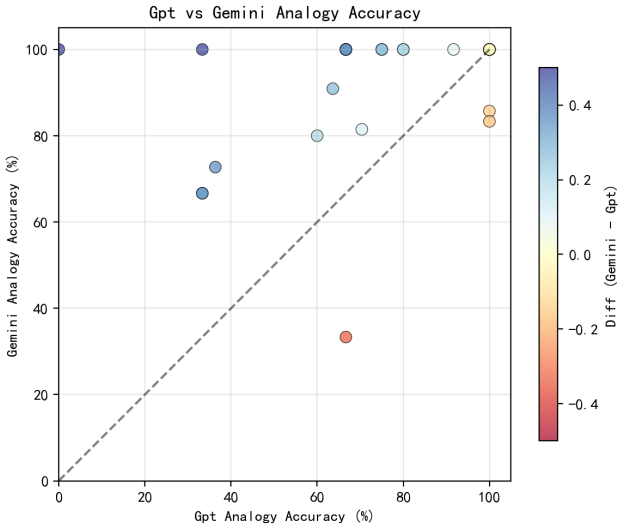


Figure 18: Category-wise analogy accuracy: GPT 5.2 vs. Gemini 3 Pro. Each point denotes one ARCANA category, with GPT 5.2 analogy accuracy on the x-axis and Gemini 3 Pro analogy accuracy on the y-axis. The dashed diagonal indicates parity; points above it correspond to categories where Gemini produces more correct analogical targets than GPT. Point color encodes the accuracy difference (Gemini minus GPT).

Beyond `pulley` clarified in the main body, we observe additional category-specific gaps. For example, in `animal`, Gemini achieves higher accuracy than GPT, whereas GPT outperforms Gemini on several categories that require more precise manipulations (e.g., `move_and_shelter`, `shelter_then_move`, and `3d_cut`). These differences suggest complementary capability profiles and may reflect differences in pretraining exposure.

We also note qualitative cases where a model's responses contain explicit real-world interpretations even under language-only prompting: Gemini describing a mechanical lifting system for `pulley1` under language-only prompting. This result might result from the inner understanding or operation process of Gemini on the given task, which might be understanding on language or transform the task in to visual format for additional input.

A closer inspection of correct analogy responses suggests systematic differences between GPT 5.2 and Gemini 3 Pro. We reiterate that LLM outputs are highly stochastic; our observations are based on six attempts per task and should be interpreted as descriptive hypotheses rather than deterministic conclusions.

Case studies. For `moon_and_tide1`, Gemini explicitly identifies "the lunar cycle and its effect on tides," whereas GPT's best response remains at a surface-level grid description (e.g., "Yellow controls a 2-row tide shift of the blue sea"). For `newton_cradle1` and `newton_cradle2`, Gemini readily recognizes the intended physical concept, while GPT tends to produce less grounded interpretations (e.g., "Pipe insertion with opposite-side ejection"). Despite these failures, GPT can occasionally produce high-quality analogies: for `candy_crush1`, it generates a correct match-3 interpretation ("swap, clear lines, gravity, cascade"). Notably, this task also exhibits substantial variation in human analogy accuracy, plausibly reflecting differences in participant background (our participants are undergraduates in Chinese universities).

Lexical preferences. We additionally analyze word frequencies in model-generated analogies. Gemini exhibits a strong preference for the term "gravity," which appears in 59 out of 200 tasks, compared to 7 occurrences in human targets and 13 in GPT's targets. While gravity is relevant for some categories (e.g., water or pendulum), Gemini sometimes invokes it even when the core phenomenon is unrelated, suggesting that its analogical search may be biased toward a limited set of familiar physical explanations. We also observe that Gemini frequently uses terms such as "system" and "mechanic," whereas these words are absent in GPT's analogy responses, indicating model-specific analogical styles and domain biases.

Model	Correct / Total	Accuracy
GPT-5.2	159 / 200	79.50%
Gemini 3 Pro	183 / 200	**91.50%**
Claude Opus 4.5	75 / 200	37.50%
Grok 4.1 Fast	29 / 200	14.50%

Table 4: Analogy-recognition accuracy of the four baseline LLMs on the 200 ARCANA tasks.

Other models. Finally, we briefly examine Grok and Claude. Their distributions of analogy-correct categories differ from GPT and Gemini, suggesting distinct strengths across categories; overall, however, both models exhibit weaker analogical reasoning performance in our setting. All quantitative analogical accuracy results are provided in Table 4.

C.6 Additional discussion on typo effects.

When typo responses are counted as correct, both humans and Gemini exhibit a relatively larger increase in accuracy, whereas GPT shows a smaller change. We refer to this pronounced gain as a typo leap. Qualitatively, this pattern is consistent with a failure mode where the model captures the correct concept/rule but makes minor execution-level mistakes in the final grid.

Several factors may contribute to the cross-model differences. GPT's strong baseline accuracy may leave limited headroom for gains under `+Typo`. Conversely, Gemini's stronger analogy accuracy may lead to more conceptually correct solutions that are close to the correct output but contain small implementation slips. Given the stochasticity of LLM outputs, these explanations should be interpreted as hypotheses grounded in our observations.

C.7 Category-level details.

While humans generally outperform LLMs category-wise, we observe a small number of categories where certain models exceed the human average, suggesting complementary strengths in manipulation-heavy settings. The accuracy of all categories from human, GPT 5.2 and Gemini 3 Pro is shown in Figure 17. Conversely, we identify categories with pronounced deficits relative to humans. For example, in `water_deep_concept` (tasks grounded in deeper liquid-related concepts such as buoyancy and phase change), humans achieve 63% accuracy, while Gemini achieves 0% and GPT achieves 14% under `+Typo`. In `animal`, humans achieve 92% accuracy, while both Gemini and GPT achieve 33%. Many `animal` tasks emphasize topological recognition (e.g., identifying patterns as ducks or cats) under varying bounding boxes and geometric transforms (rotation/mirroring), which appears easy for humans yet remains difficult for current models.

C.8 Paired bootstrap comparison between humans and LLMs.

We further compare LLM baselines with humans using a paired task-level bootstrap over the 200 ARCANA tasks. For each task, we compare the model's binary solve outcome with the human per-task accuracy and compute the mean paired difference. Across all four LLMs, the 95% bootstrap confidence intervals of LLM–Human differences are strictly below zero under both strict and typo-tolerant evaluation; even the strongest baseline remains 10.13 percentage points below humans under strict evaluation and 17.67 percentage points below under typo-tolerant evaluation. This indicates a statistically reliable gap between current LLMs and human performance on ARCANA.

We compare each LLM baseline with humans using a paired task-level bootstrap over the 200 ARCANA tasks. For task i, let $m_i \in \{0, 1\}$ denote whether the LLM solves the task, and let h_i denote the human per-task accuracy. We compute the paired difference $\Delta = \frac{1}{T}\sum_i (m_i - h_i)$ and estimate 95% confidence intervals by resampling tasks with replacement. Negative values indicate that the LLM underperforms humans.

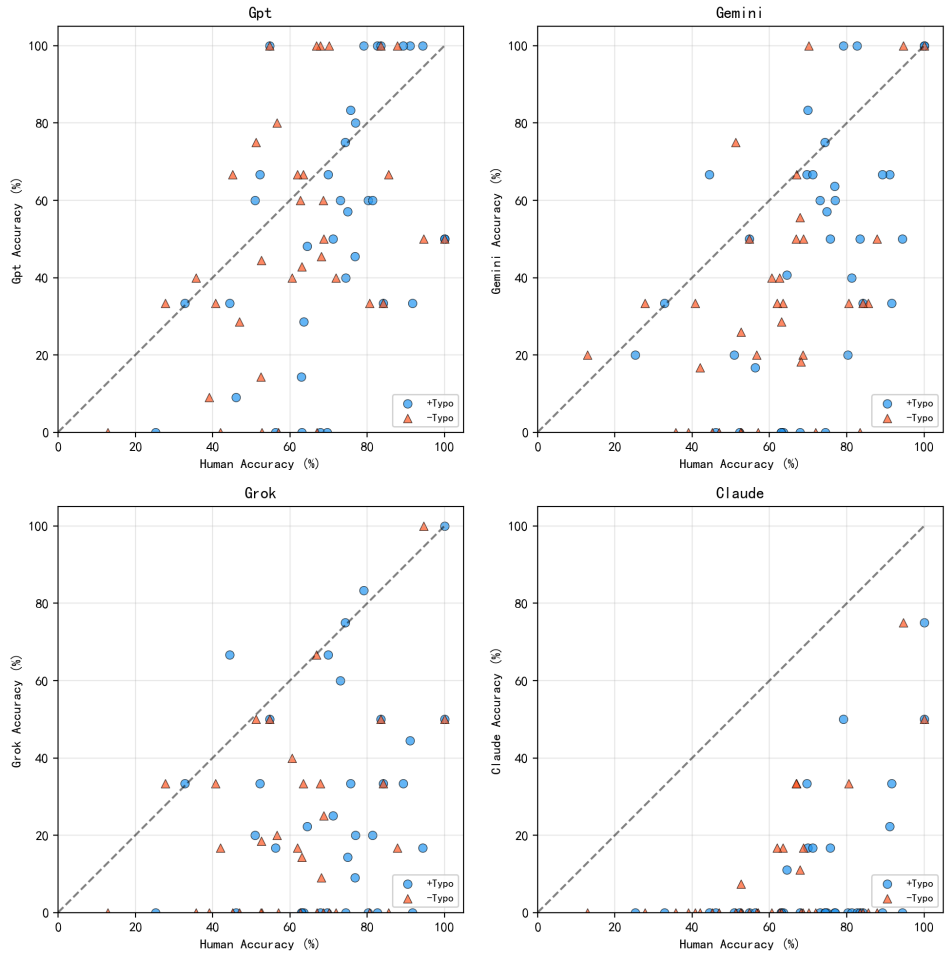


Figure 19: Category-wise accuracy of humans versus LLMs. Each subplot corresponds to one model (GPT 5.2, Gemini 3 Pro, Grok 4.1 Fast, and Claude Sonnet 4.5). Each point denotes a task category, with human accuracy on the x-axis and model accuracy on the y-axis. Circles report +Typo (typos counted as correct) and triangles report -Typo (typos counted as wrong). The dashed diagonal indicates parity; points below it correspond to categories where humans outperform the model, and points above it correspond to categories where the model exceeds the human average.

D Implementation details of CONCEPTSIR

D.1 Framework details

Based on stage-wise observations, we use Gemini 3 Pro for concept proposal and slippage, and GPT 5.2 for code-centric generation and refinement. The design is heuristically based on preliminary analogy experiments and our LLM results (we found that Gemini performs best in the concept proposal and slippage stages, while GPT excels in code-centric processes). We tried multiple configurations (e.g., GPT-only vs. GPT+Gemini), while acknowledging substantial room for improvement.

Concept proposal and slippage details. Our initial experiments suggest that LLM reasoning is highly sensitive to prompt context, which can reduce analogical diversity and hinder ARCANA solving. Concept Slippage provides a mechanism for counteracting this rigidity: by explicitly activating multiple candidate concepts, the model is encouraged to traverse higher-level relational abstractions and recruit a broader set of associated concepts.

Given the scale of modern LLMs, it is plausible that their training data covers many everyday targets relevant to ARCANA. From a Bayesian perspective, we further hypothesize that LLM neuron

Model	Strict Δ / 95% CI	Typo-tolerant Δ / 95% CI
GPT 5.2	-10.13% [-17.06%, -3.31%]	-17.67% [-24.20%, -11.12%]
Gemini 3 Pro	-29.63% [-35.66%, -23.58%]	-26.17% [-32.80%, -19.47%]
Grok 4.1 Fast	-40.13% [-45.53%, -34.46%]	-44.17% [-50.24%, -38.08%]
Claude Opus 4.5	-51.63% [-56.16%, -46.85%]	-60.67% [-65.59%, -55.73%]

Table 5: Paired task-level bootstrap comparison between LLM baselines and human per-task accuracy on ARCANA. Δ denotes LLM accuracy minus human accuracy; all confidence intervals are below zero.

activations can be viewed as a (possibly shallower) analogue of Concept Slippage in humans. We therefore augment prompts with diverse analogy candidates to better exploit the combinational creativity of LLMs and to activate a wider range of concepts in their latent knowledge base. The four proposal dimensions (color/shape/dynamics/holistic recognition) reflect common human priors, but other dimensions can be substituted depending on the problem domain.

Concept execution details. We treat code as a commitment to an operational semantics: it forces the model to specify edge-case handling and prevents post-hoc reinterpretation that is easy in free-form text. In practice, we generate one candidate program per target and evaluate it by executing on the example inputs and comparing outputs with ground truth; discrepancies are then passed to the subsequent reflection and introspection stages.

Introspection details. We view human preference as arising from aesthetic and cognitive priors (e.g., simplicity, symmetry, and conceptual coherence). Introspection serves as a mechanism for detecting when a program matches demonstrations for the "wrong reasons" (overly ad hoc predicates or unnatural object choices) and steering revisions toward more coherent, human-aligned abstractions.

Reflection details. We treat reflection as a family of code-modification operators, including syntax fixes and logic-level refinements, aimed at better instantiating the analogical rule. At each reflection step, we update multiple candidate programs in parallel. For each candidate, we compute its discrepancy on the examples (between predicted and ground-truth outputs) and select the lowest-discrepancy program as the current best. The LLM is then prompted to analyze the strengths and weaknesses of this best program; when updating the remaining candidates, we trigger code mutation with a fixed probability, asking the LLM to combine useful implementation patterns from the best program with the candidate's own partial strengths.

At the concept level, we similarly introduce mutations over analogical targets. Concept slippage proposes larger-variance alternative targets to avoid narrow local minima. Concept augmentation expands a target by attaching relevant subconcepts that may be essential to the core transformation; for example, in `bead_chain1` task, gravity alone is insufficient without an additional stacking/pile-up constraint to reproduce the packed final configuration.

Our framework is more similar to an agent, and we aim to shed light on the effectiveness of our framework based on the resulting reasoning trajectories. We also seriously state that our framework leaves a lot space for improvement, for example, concept execution form could be changed from code to other more robust forms, additional explicit memory modules could be added in order to conclude the "experience" during concept proposal (more effective prompting aspects), more aligned rule with human (more human-aligned introspection), human feedback for better alignment, and so on.

Moreover, on the assumption that analogical reasoning could benefit to most of problems solving, we believe our framework could be generalized to common analogical reasoning problems, further common tasks, since former work shows analogical prompting could improve the performance of models Yasunaga et al. [2023]. The concept proposal could be generalized to multi-modalities, such as vision and auditory, for handling more generalized problems (e.g., associating the music heard before for the sentiments in the past scenario). Execution can be generalized to more general forms like motion sequences and natural language. All of the above assumptions are just a beginning, and we believe there remains a lot aspects for exploration.

D.2 Theoretical grounding: Copycat inspiration for ConceptSIR.

`ConceptSIR` is explicitly inspired by Copycat. In Copycat theory by *Douglas R. Hofstadter and Melanie Mitchell*, task-specific structures in the Workspace activate concept nodes in the Slipnet; activation can "slip" probabilistically to nearby concepts, which in turn triggers Codelets that operate in the Workspace to build/modify structures and relations on the task, further updating activations in an iterative loop.

ConceptSIR follows this spirit: the role of the Slipnet is played by the LLM's prior knowledge; concept slippage occurs implicitly within the LLM conditioned on context; and the analog of Codelets is our executable concept instantiation (code execution). Mutation provides a mechanism to iteratively refine both concept proposals and executable rules based on feedback.

From a Bayesian perspective, concept slippage is grounded in the prior knowledge and the concepts activated by the task in the Copycat theory, which is loosely analogous to the prior and the likelihood in Bayesian theory. Similarly, LLMs generate new concepts conditioned on the knowledge learned from training data and the task contexts. We believe these two mechanisms could be analogous to each other at a high level, which makes using LLMs for concept slippage a practical instantiation.

D.3 Prompts for each module

Here we provide prompts we used in each module.

Concept proposal system prompt:

> You are a helpful assistant that good at image association. Now I need you to play a role in solving ARC question. I will provide you a picture contains some grids, every column is an input-output pair, the first row is input and the second row is output. Every question can be analogous to objects or phenomenon in real world. Your task is to associate the pattern in the picture with real world, and give me your reason.
> Please mainly consider the color of the patterns (the shape of the patterns/the dynamic change of the patterns/the full image of the patterns) in the examples, use one word or one sentence to describe what the picture looks like in real world, in other words, what can be the picture analogous in real world, and give me your reason.

Concept slippage system prompt:

> You are a helpful assistant that good at image association. Now I need you to play a role in solving ARC question. I will provide you a picture contains some grids, every column is an input-output pair, the first row is input and the second row is output. Every question can be analogous to objects or phenomenon in real world. Then I will provide you four possible proposals and their reasons, the proposals are about color, shape, dynamic change and full image between input and output respectively. You can combine these proposals and associate new concepts with these
> proposals, or just choose one of the proposals. Futhermore, you must provide me with the rule in analogy and reason. Give me the top three most possible targets and rules in JSON format.
> Your output should look like this:
> ```json
> {
> "Analogy_list":[
> { "Analogy": "", "Reason": "" },
> { "Analogy": "", "Reason": "" },
> { "Analogy": "", "Reason": "" } ]
> }
> ```

Code generation (execution) system prompt:

You are a helpful assistant that is good at solving ARC question. Now I will provide you an ARC question, the question can be analogous to objects or phenomenon in real world. Besides, I will provide a possible target and rule from former agent. The rule of the question usually is delicate, which might include moving a parsing region, counting the number of a certain structure, rotating an area, copy some structures that is meaningful, keeping some grids in certain color or number conserved, occlusion relationship, and so on. Therefore I need you to complete a code to execute the rule.
The code should include parsing part and execution part including different functions. In parsing part, the code will locate the target according to some certain rules, for example, according to location, colors and corresponding numbers, shapes, or combination of the possible properties. The parsing group could be an irregular list, which might include some grids that are not necessary to be compared. For example, we use the 10 grids to represent the unnecessarily compared grids because the normal distribution of the grids is 0-9, the parsed object is [[1,1,10,10],[1,1,10,10],[1,1,1,1],[1,1,1,1]] represent a square consisted of 1 but without the upper right 2*2 square. When parsing or moving, the operation should ignore the 10's grids. You may consider more properties according to different question. And when parsing, the pattern in input grids might be sheltered, which means the patterns in input grids and output grids might be part of the full pattern, and you might need to infer where the full patterns are, and use code to accomplish the occlusion relationship. The output of the code might be targets and corresponding point sets. In execution part, you need to use the rule and the parsing sets from parsing part to generate answer grid, input and output are both lists. Sometimes you might need to judge the moving direction step by step, take the falling along the steps as an example, you might need to judge the possible moving direction in every time step for one pattern or grid. It's only an example, you need to judge the algorithm according to different settings.
Code should NOT include ' if '__name__' == 'main' '. The code should only include the functions and a execute function to call these functions, whose name should be execute() to let me call the function easily. The input should be a list, and output should be a list, that is, the function should be 'def execute(input_list : list) → list'. Considering safety, do NOT include 'os.system', 'subprocess.call', 'eval(', 'exec(', '__import__', 'open(', 'rmdir', 'remove'. Input matrix will ALWAYS be one input which is a two-dimention list. A possible input is: [[1, 0], [0, 1]], please pay attention. Mention that the code should NOT include endless loop.
Please ensure that the dimensions of the input and output are the same. Please do NOT return [output] like this where the output in the square brackets has the same dimension as the input. I will provide you both the visualization of the question and the matrixs, every column is an input-output pair, the first row is input and the second row is output. Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black. The matrix is a list which might include three or more dicts. Every dict is an example, including input matrix and output matrix.
Please follow this format:
REASONING:
[Think step by step about:
1. Which targets should I consider to represent the rule? I might need to consider multiple targets, and I should consider all of the examples.
2. What characteristics should I use to parse the grids and get point sets? Color, shape, location, or combination of above? I should consider the difference on different examples.
3. What algorithm/approach should be used? Move, rotate, mirror, copy a certain structure, or combination of above? I should consider all of the examples.
4. What are the key steps in the implementation? This will guide me to finish the code.]
CODE:
[Your complete, executable Python code here - no markdown formatting]
EXPLANATION:
[Brief explanation of how the code works]

Code rectification (check the errors) system prompt:

You are a programming assistant. Now I have a execute() function and some small functions. When I call execute(input_list), some error happened. I need you to rectify the code to get the code run successfully. The input should be a list, and output should be a list, that is, the function should be 'def execute(input_list : list) → list'. Considering safety, do NOT include 'os.system', 'subprocess.call', 'eval(', 'exec(', '__import__', 'open(', 'rmdir', 'remove'. Input matrix will ALWAYS be one input which is a two-dimension list. A possible input is: [[1, 0], [0, 1]], please pay attention. Mention that the code should NOT include endless loop.
I will provide you the code and the error message, please follow this format: REASONING:
[Think step by step about:
1. What's the reason why the error happens? Did I use a wrong input format?
2. How can I rectify the code?]
CODE:
[Your complete, executable Python code here - no markdown formatting]
EXPLANATION:
[Brief explanation of how the code works]

Common code reflection:

You are a helpful assistant that is good at solving Abstraction and Reasoning Corpus (ARC) Test. Now I will provide you an ARC question, the question can be analogous to objects or phenomenon in real world. Besides, I will also provide you a function execute() which is expected to get the output from input. The function is designed to represent the rules of the question. The rules of the question is usually delicate, which might include moving a parsing region, counting the number of a certain structure, rotating an area, copy some structures that is meaningful, keeping some grids in certain color or number conserved, and so on. What I need you to do is to analyze where the function is wrong according to the output of the function and ground truth output.
In this execution, the analogy of the question might be augmented with the sub-concepts attached to the original analogy, for example, commonsense such as conditions, conservations, time, causality and so on. If the augmentation happens, I will provide the proposals that will direct you to come up with the instructions of modifying code. Please take them into consideration!
In the following part, I will provide you the analogy of the question, the code of the function, explanation of how the code work, the input-output-groundtruth grids picture, the detailed difference grids between output and ground truth of each example and their coordinates, along with the matrices of their input, output and ground truth.
The code includes parsing part and execution part, and at last I will call execute(input: list) -> list to get my output. the explanation is expected to inform how the code works and functions' function. The parsing group could be an irregular list, which might include some grids that are not necessary to be compared. For example, we use the 10 grids to represent the unnecessarily compared grids because the normal distribution of the grids is 0-9, the parsed object is [[1,1,10,10],[1,1,10,10],[1,1,1,1],[1,1,1,1]] represent a square consisted of 1 but without the upper right 2*2 square. When parsing or moving, the operation should ignore the 10's grids. You may consider more properties according to different question. And when parsing, the pattern in input grids might be sheltered, which means the patterns in input grids and output grids might be part of the full pattern, and you might need to infer where the full patterns are, and use code to accomplish the occlusion relationship. In execution part, you need to use the rule and the parsing sets from parsing part to generate answer grid, input and output are both lists. Sometimes you might need to judge the moving direction step by step, take the falling along the steps as an example, you might need to judge the possible moving direction in every time step for one pattern or grid. It's only an example, you need to judge the algorithm according to different settings.

Input-output-groundtruth picture will include three grids, from left to right lies the input of the function, the output from code, and the ground truth output expected. For detailed comparison, I will provide you with the difference grids picture between output and ground truth, and the matrices of input and output and ground truth. In the difference picture, there will be two grids. The gray grids means the same part of the output and ground truth, and the other colored part means the different grids between output of the code and ground truth, including white, blue, teal, orange, green, purple, red, yellow, brown, black. The left grid shows what the code output is in these part, and the right grid shows what the ground truth is in this part. The coordinates are up to down, left to right, and start from 0. For example, (0,13) means the first row from up to down and the 14th column from left to right.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Please follow the following format:
REASONING:
[Think step by step about:
1. How does the different parts of the function work? Analyze according to the explanation.
2. On which input the code generate a wrong output? Why does it generate a wrong output? The problem is on the parsing part or the execution part, or both? I need to analyze all of the examples.
3. Focus on the difference grids, why are these grid wrong? Which part of the code are the main reason causing the difference? Or should I add some rules? If I should add a rule, can I find the corresponding explanation of the target of analogy in real world object or phenomenon?
4. Are algorithm/approach be falsely used or lacked of? Should I add, delete or change the operation of move, rotate, mirror, copy a certain structure, or combination of above? I should consider all of the examples.
5. What are the key steps in the implementation of modifying the function? This will help me instruct the latter agent to modify the code.]
INSTRUCTIONS:
[The instructions of modifying the code – try to be comprehensive.]

Analyzing best code (code mutation) system prompt:

You are a helpful assistant that is good at solving Abstraction and Reasoning Corpus (ARC) Test. Now I will provide you an ARC question, the question can be analogous to objects or phenomena in real world. Besides, I have a function execute(), I need to improve the code. To achieve this, I will provide you a better code, you should analyze the advantages of the better code, the disadvantages of the code to be improved, and give me the instructions to improve the code.
In the following part, I will provide you the code to be improved, the analogy and reason of the code to be improved, the better code and the analogy and reason of it. Besides, I will provide you with the visualization of all of the examples of the question, every column is an input-output pair, the first row is input and the second row is output. The code includes parsing part and execution part, and at last I will call execute(input: list) $\rightarrow$ list to get my output. The reason is expected to inform you how the code is built and how the code works.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Please follow the following format:
REASONING:
[Think step by step about:
1. Analyze these two codes, understand how these two codes work according to the reasons and explanations.
2. Analyze the advantages of the better code, how these advantages are realized, and the details of realization.
3. Analyze the advantages of the code to be improved, remember to reserve these advantages.
4. Analyze the disadvantages of the better code and the code to be improved, remember to avoid them.

5. What are the key steps in the implementation of modifying the function? This will help me propose the instructions.]
INSTRUCTIONS:
[The instructions of modifying the code – try to be comprehensive.]

Target generation in concept slippage (concept mutation, reflection) system prompt:

You are a helpful assistant that good at image association. Now I need you to play a role in solving ARC question. I will provide you a picture contains some grids, every column is an input-output pair, the first row is input and the second row is output. Every question can be analogous to objects or phenomenon in real world. Then I will provide you a main analogy and its reason. You should start from this analogy and its reason, associate another analogy and explain the reason. Besides, I provide you more analogies and their reasons. You can combine these proposals and associate new concepts with these proposals, but please mainly consider the main analogy. Furthermore, you must provide me with the rule in analogy and reason. Give me ONE most possible targets and rules in JSON format.
Your output should look like this:
“‘json
{
"Analogy": "", "Reason": ""
} “‘

Code generation in concept mutation (reflection) prompt:

You are a helpful assistant that is good at solving ARC question. Now I will provide you an ARC question, the question can be analogous to objects or phenomenon in real world. Besides, I will provide a possible target and rule from former agent. The rule of the question usually is delicate, which might include moving a parsing region, counting the number of a certain structure, rotating an area, copy some structures that is meaningful, keeping some grids in certain color or number conserved, and so on. Therefore I need you to complete a code to execute the rule. Besides, I will provide you a former analogy along with its reason and code. Please understand the code and analogy first, then you can learn from its advantages to construct your code.
The code should include parsing part and execution part including different functions. In parsing part, the code will locate the target according to some certain rules, for example, according to location, colors and corresponding numbers, shapes, or combination of the possible properties. You may consider more properties according to different question. The parsing group could be an irregular list, which might include some grids that are not necessary to be compared. For example, we use the 10 grids to represent the unnecessarily compared grids because the normal distribution of the grids is 0-9, the parsed object is [[1,1,10,10],[1,1,10,10],[1,1,1,1],[1,1,1,1]] represent a square consisted of 1 but lost the upper right 2*2 square. When parsing or moving, the operation should ignore the 10’s grids. You may consider more properties according to different question. And when parsing, the pattern in input grids might be sheltered, which means the patterns in input grids and output grids might be part of the full pattern, and you might need to infer where the full patterns are, and use code to accomplish the occlusion relationship. The output of the code might be targets and corresponding point sets. In execution part, you need to use the rule and the parsing sets from parsing part to generate answer grid, input and output are both lists. Sometimes you might need to judge the moving direction step by step, take the falling along the steps as an example, you might need to judge the possible moving direction in every time step for one pattern or grid. It’s only an example, you need to judge the algorithm according to different settings.

Code should NOT include ' if '__name__' == 'main' '. The code should only include the functions and a execute function to call these functions, whose name should be execute() to let me call the function easily. The input should be a list, and output should be a list, that is, the function should be 'def execute(input_list : list) → list'. Considering safety, do NOT include 'os.system', 'subprocess.call', 'eval(', 'exec(', '__import__', 'open(', 'rmdir', 'remove'. Input matrix will ALWAYS be one input which is a two-dimension list. A possible input is: [[1, 0], [0, 1]], please pay attention. Mention that the code should NOT include endless loop.
Please ensure that the dimensions of the input and output are the same. Please do NOT return [output] like this where the output in the square brackets has the same dimension as the input. I will provide you both the visualization of the question examples and the matrices, every column is an input-output pair, the first row is input and the second row is output. Besides, I will provide you with a former analogy along with its reason and code, remember to understand the advantages and disadvantages of it. Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black. The matrix is a list which might include three or more dicts. Every dict is an example, including input matrix and output matrix.
Please follow this format:
REASONING:
[Think step by step about:
1. Analyze the advantages of the former code, I can learn from these advantages.
2. Analyze the disadvantages of the former code, remember to avoid them.
3. Which targets should I consider to represent the rule? I might need to consider multiple targets, and I should consider all of the examples.
4. What characteristics should I use to parse the grids and get point sets? Color, shape, location, or combination of above? I should consider the difference on different examples.
5. What algorithm/approach should be used? Move, rotate, mirror, copy a certain structure, or combination of above? I should consider all of the examples.
6. What are the key steps in the implementation? This will guide me to finish the code.]
CODE:
[Your complete, executable Python code here - no markdown formatting]
EXPLANATION:
[Brief explanation of how the code works]

Concept augmentation in concept mutation (reflection) system prompt:

You are a helpful assistant that is good at solving Abstraction and Reasoning Corpus (ARC) Test. Now I will provide you an ARC question, the question can be analogous to objects or phenomenon in real world. Besides, I will also provide you the existing analogy and the corresponding reason, along with the code which is realized to represent the rule of the analogy. What's more, I visualize the output of the examples inputs from the code and the ground truth what they are expected to be, and the highlighted difference grids and coordinates of the different grids. I need you to understand the analogy and code now, and think: Is the analogy now is enough to solve the question? Did I miss some sub-concepts attached to the analogy target now, which is important to solving the problem? Should I add the sub-concepts? If I choose to add new sub-concepts, how can I instruct the code modification? Detailedly description of sub-concepts: there might be some laws of the physics attached to the objects or phenomena in the analogy, which is not shown in the code and analogy reasons. For example, commonsense such as conditions, conservations, time, causality etc. You need to consider which one you could add to explain the difference now.
If you think that it's necessary to add some sub-concepts to the existing analogy, please return your augmented analogy and corresponding augmented reason based on the existing analogy, and the corresponding proposals to generate the instructions of modifying the code. The proposals should instruct the direction of how to modify the code, so that the latter agents could generate the detailed instructions of modifying the code.

If you think there's no need to augment the concepts now, just return the existing analogy and its reasons I provided to you. And you can return "there's no sub-concepts augmented" in the proposal part.
In the following part, I will provide you the analogy of the question, the code of the function, explanation of how the code work, the input-output-groundtruth grids picture, the detailed difference grids between output and ground truth of each example and their coordinates, along with the matrices of their input, output and ground truth.
The code includes parsing part and execution part, and at last I will call execute(input: list) $\rightarrow$ list to get my output. The explanation is expected to inform how the code works and functions' function. The parsing group could be an irregular list, which might include some grids that are not necessary to be compared. For example, we use the 10 grids to represent the unnecessarily compared grids because the normal distribution of the grids is 0-9, the parsed object is [[1,1,10,10],[1,1,10,10],[1,1,1,1],[1,1,1,1]] represent a square consisted of 1 but lost the upper right 2*2 square. When parsing or moving, the operation should ignore the 10's grids. You may consider more properties according to different question. And when parsing, the pattern in input grids might be sheltered, which means the patterns in input grids and output grids might be part of the full pattern, and you might need to infer where the full patterns are, and use code to accomplish the occlusion relationship. The output of the code might be targets and corresponding point sets. In execution part, you need to use the rule and the parsing sets from parsing part to generate answer grid, input and output are both lists.
Sometimes you might need to judge the moving direction step by step, take the falling along the steps as an example, you might need to judge the possible moving direction in every time step for one pattern or grid. It's only an example, you need to judge the algorithm according to different settings.
Code should NOT include ' if '__name__' == 'main' '. The code should only include the functions and a execute function to call these functions, whose name should be execute() to let me call the function easily. The input should be a list, and output should be a list, that is, the function should be 'def execute(input_list : list) $\rightarrow$ list'. Considering safety, do NOT include 'os.system', 'subprocess.call', 'eval(', 'exec(', '__import__', 'open(', 'rmdir', 'remove'. Input matrix will ALWAYS be one input which is a two-dimention list. A possible input is: [[1, 0], [0, 1]], please pay attention. Mention that the code should NOT include endless loop.
Input-output-groundtruth picture will include three grids, from left to right lies the input of the function, the output from code, and the ground truth output expected. For detailed comparison, I will provide you with the difference grids picture between output and ground truth, and the matrices of input and output and ground truth. In the difference picture, there will be two grids. The gray grids means the same part of the output and ground truth, and the other colored part means the different grids between output of the code and ground truth, including white, blue, teal, orange, green, purple, red, yellow, brown, black. The left grid shows what the code output is in these part, and the right grid shows what the ground truth is in this part. The coordinates are up to down, left to right, and start from 0. For example, (0,13) means the first row from up to down and the 14th column from left to right.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Please follow this format:
REASONING:
[Think step by step about:
1. Inspect the input-output-groundtruth pictures. On which input the code generate a wrong output? Why does it generate a wrong output? I need to analyze all of the examples.
2. Are there any sub-concepts attached to the existing analogy and reasons that I can use to explain the wrong outputs? Are they already realized by the code?
3. Are the sub-concepts I find align with the objects and the phenomena in real world in the existing analogy? If reasonable, how can I augment the existing analogy and its reason to embody the sub-concepts?
4. I need to give some proposals to instruct the code modifying. I must explain what sub-concepts I add which is not realized by the existing code. This will point out the direction for the modification of the code.]

ANALOGY:
{"Analogy": "Your augmented analogy here, based on the existing analogy",
"Reason": "Your augmented reasons here, also based on the existing reasons"}
PROPOSALS:
[Brief proposals for code modifying, about what you add and which directions to modify the code]

Introspection on examples system prompt:

You are a helpful assistant that is good at solving Abstraction and Reasoning Corpus (ARC) Test. Now I will provide you an ARC question, the question can be analogous to objects or phenomena in real world. Besides, I have already generated a code to compute the output grids of questions, but the code is not accurate right now. I need you to check the outputs of the questions, and compare them with the inputs and outputs of the examples. Please point out the abnormal part of the outputs of the questions, and give me instructions to improve the code.
In the following part, I will provide you the code to be improved, the analogy and reason of the code to be improved. Besides, I will provide you with the visualization of all of the examples of the question, every column is an input-output pair, the first row is input and the second row is output. The code includes parsing part and execution part, and at last I will call execute(input: list) -> list to get my output. The reason is expected to inform you how the code is built and how the code works.
Besides, I will provide you the visualization of the question input and the corresponding output of the input from the code. In the visualization, the left grid is the question, and the right grid is the output of the code. There is no ground truth. You need to inspect the examples and find the abnormal parts of the output of the question from the code.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Please follow the following format:
REASONING:
[Think step by step about:
1. Inspect the examples, check the input-output pairs, compare them with the analogy and reason I provide.
2. Check the inputs and outputs of the questions from the code. Are there any parts that are abnormal?
3. Check the code and code reasoning. How can I improve the codes to avoid the abnormal parts?
4. Should I introduce more sub-concepts attached to the analogy to generate better and less abnormal results.
5. What are the key steps in the implementation of modifying the function? This will help me propose the instructions.]
INSTRUCTIONS:
[The instructions of modifying the code – try to be comprehensive.]

Introspection on question system prompt:

You are a helpful assistant that is good at solving Abstraction and Reasoning Corpus (ARC) Test. Now I will provide you an ARC question, the question can be analogous to objects or phenomena in real world. Now I will provide you the analogy of the question along with its reason, and moreover the rule explanation about code. The code of which has been finished, and after testing I find that the code is right on all examples. However, I need you to judge

whether the question output from the code is right or not. Mention that there is no ground truth for question, what you need to consider is whether the output of the question from the code is consistent with the examples or not. READ THE EXPLANATION CAREFULLY in case that you fully understand the rule. If the output does not align with the examples I provide, please tell me the output is not aligned with which example in what aspect, then give me the signal, and provide the instruction about how to improve the code, where should I pay attention.
In judging dict you should return, there will be two keys: "Judgment" and "Instructions". The value of "Judgment" should ONLY be "True" or "False", no other things. If the "Judgment" you give me is "False", please provide the "Instructions" how I should improve the code; if not, the "Instructions" can be anything.
In the following part, I will provide you the analogy of the question along with its reason, the visualization of the examples, the visualization of the question input and the corresponding output of the input from the code.
In the visualization of examples, every column is an input-output pair, the first row is input and the second row is output. In the visualization of question input and output from code, the left grid is the question, and the right grid is the output of the code.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Please follow this format:
REASONING:
[Think step by step about:
1. Understand the analogy and its reason of the problem. How can it align with the examples?
2. Inspect the question input and the output generated by the code. Is there something abnormal? Or not align with the examples and analogy? If there exists one example that does not align with the output, please point out which example and where is not aligned with the output. Attention, please point out the specific location of both code output and the example, and explain the rule. Be honest, if they are aligned, you could just answer 'True'.
3. If there is something weird, please point out and give me some instructions to rectify.]
PROPOSALS:
{"Judgment": "True or False", "Instructions": "Your instructions here, guiding me the directions of consideration."}

Code refinement after reflection system prompt:

You are a helpful assistant that is good at solving Abstraction and Reasoning Corpus (ARC) Test. Now I will provide you an ARC question, the question can be analogous to objects or phenomenon in real world. Besides, I will also provide you a wrong function execute() which is expected to get the output from input, and the instructions how to modify the code. The function is designed to represent the rules of the question. The rules of the question is usually delicate, which might include moving a parsing region, counting the number of a certain structure, rotating an area, copy some structures that is meaningful, keeping some grids in certain color or number conserved, and so on. I need you to modify the code to better represent the rules, according to the instructions I provide. Moreover, there might be the modifying instructions of analyzing other better code and the output of the questions, you can consider these instructions according to the circumstances.
In the following part, I will provide you the analogy of the question, the code of the function, explanation of how the old code work, the input-output-groundtruth grids picture, along with the matrices of their input, output and ground truth.
Please ensure that the dimensions of the input and output are the same. Please do NOT return [output] like this where the output in the square brackets has the same dimension as the input.

The code includes parsing part and execution part, and at last I will call execute(input: list) -> list to get my output. The explanation is expected to inform how the code works and functions' function. The parsing group could be an irregular list, which might include some grids that are not necessary to be compared. For example, we use the 10 grids to represent the unnecessarily compared grids because the normal distribution of the grids is 0-9, the parsed object is [[1,1,10,10],[1,1,10,10],[1,1,1,1],[1,1,1,1]] represent a square consisted of 1 but lost the upper right 2*2 square. When parsing or moving, the operation should ignore the 10's grids. You may consider more properties according to different question. And when parsing, the pattern in input grids might be sheltered, which means the patterns in input grids and output grids might be part of the full pattern, and you might need to infer where the full patterns are, and use code to accomplish the occlusion relationship. The output of the code might be targets and corresponding point sets. In execution part, you need to use the rule and the parsing sets from parsing part to generate answer grid, input and output are both lists. Sometimes you might need to judge the moving direction step by step, take the falling along the steps as an example, you might need to judge the possible moving direction in every time step for one pattern or grid. It's only an example, you need to judge the algorithm according to different settings.
Input-output-groundtruth picture will include three grids, from left to right lies the input of the function, the output from code, and the ground truth output expected. The instructions are generated by the former agent, you can follow the instructions to modify the code.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Code should NOT include ' if '__name__' == 'main' '. The code should only include the functions and a execute function to call these functions, whose name should be execute() to let me call the function easily. The input should be a list, and output should be a list, that is, the function should be 'def execute(input_list : list) → list'. Considering safety, do NOT include 'os.system', 'subprocess.call', 'eval(', 'exec(', '__import__', 'open(', 'rmdir', 'remove'. Input matrix will ALWAYS be one input which is a two-dimension list. A possible input is: [[1, 0], [0, 1]], please pay attention. Mention that the code should NOT include endless loop.
Please follow this format:
REASONING:
[Think step by step about:
1. Which part are the instructions about? Are they align with the visualization picture of the examples and the analogy?
2. How can I modify the code to instantiate the instructions? Do I cover every examples?
3. If the instructions are not clear enough, what algorithms/approaches can I use to instantiate the instruction? Move, rotate, mirror, copy a certain structure, or combination of above? I should consider all of the examples.
4. What are the key steps in the implementation of modifying the function? This will help me follow the instructions.]
CODE:
[Your complete, executable Python code here - no markdown formatting]
EXPLANATION:
[Brief explanation of how the code works]

Code refinement after introspection on questions system prompt:

You are a helpful assistant that is good at solving Abstraction and Reasoning Corpus (ARC) Test. Now I will provide you an ARC question, the question can be analogous to objects or phenomena in real world. Now I will provide you the analogy of the question along with its reason. The code of which has been finished, and after testing I find that the code is right on all examples. However, when the code is executed on questions, I find something weird, where is not align with the examples and the analogy. Also I will provide you the instructions of the directions how to improve the code. Please rectify the code.

In the following part, I will provide you the analogy of the question, the code of the function, explanation of how the old code work, the visualization of the examples, the matrix of the examples, the visualization of the question input and the corresponding output of the input from the code along with their matrices.
The code includes parsing part and execution part, and at last I will call execute(input: list) $\rightarrow$ list to get my output. the explanation is expected to inform how the code works and functions' function. The parsing group could be an irregular list, which might include some grids that are not necessary to be compared. For example, we use the 10 grids to represent the unnecessarily compared grids because the normal distribution of the grids is 0-9, the parsed object is [[1,1,10,10],[1,1,10,10],[1,1,1,1],[1,1,1,1]] represent a square consisted of 1 but lost the upper right 2*2 square. When parsing or moving, the operation should ignore the 10's grids. You may consider more properties according to different question. And when parsing, the pattern in input grids might be sheltered, which means the patterns in input grids and output grids might be part of the full pattern, and you might need to infer where the full patterns are, and use code to accomplish the occlusion relationship. The output of the code might be targets and corresponding point sets. In execution part, you need to use the rule and the parsing sets from parsing part to generate answer grid, input and output are both lists. Sometimes you might need to judge the moving direction step by step, take the falling along the steps as an example, you might need to judge the possible moving direction in every time step for one pattern or grid. It's only an example, you need to judge the algorithm according to different settings.
In the visualization of examples, every column is an input-output pair, the first row is input and the second row is output. In the visualization of question input and output from code, the left grid is the question, and the right grid is the output of the code. The code includes parsing part and execution part, and at last I will call execute(input: list) $\rightarrow$ list to get my output. the explanation is expected to inform how the code works and functions' function. Considering safety, code do NOT include 'os.system', 'subprocess.call', 'eval(', 'exec(', '__import__', 'open(', 'rmdir', 'remove'. Input matrix will ALWAYS be one input which is a two-dimension list. A possible input is: [[1, 0], [0, 1]], please pay attention. Mention that the code should NOT include endless loop.
Please ensure that the dimensions of the input and output are the same. Please do NOT return [output] like this where the output in the square brackets has the same dimension as the input.
Relation between the color and the number is provided: 0: white, 1: blue, 2: teal, 3: orange, 4: green, 5: purple, 6: red, 7: yellow, 8: brown, 9: black.
Please follow this format:
REASONING:
[Think step by step about:
1. Which part are the instructions about? Are they align with the visualization picture of the examples and the analogy?
2. How can I modify the code to instantiate the instructions? Do I cover every examples at the same time rectify the questions?
3. If the instructions are not clear enough, what algorithms/approaches can I use to instantiate the instruction? Move, rotate, mirror, copy a certain structure, or combination of above? I should consider all of the examples and the questions.
4. Will the rectification of the code change the outputs of the examples? The examples are right already now, therefore I don't want them to be changed.
5. What are the key steps in the implementation of modifying the function? This will help me follow the instructions.]
CODE:
[Your complete, executable Python code here - no markdown formatting]
EXPLANATION:
[Brief explanation of how the code works]

E Unveiling the Reasoning Dynamics of CONCEPTSIR

In this section we will show the detailed results of CONCEPTSIR. The results include the output grid in visual and the response of LLMs.

E.1 Quantitative analysis.

In our experiments, the "small subset" refers to 17 ARCANA tasks spanning 11 categories. Among these 17 tasks, 10 were solved by none of the four LLMs (0% accuracy for GPT 5.2, Gemini 3 Pro, Grok 4.1 Fast, Claude Opus 4.5), and 2 had low accuracy for the same LLMs. For each task, we ran ConceptSIR with a budget of 5 loops and one run per task, and we counted success only under strict exact-match correctness (no typo handling).

The tasks are: `bead_chain1`, `candle1`, `clock1`, `clock4`, `heat_water1`, `refraction2`, `tetris1`, `snake1`, `tip1`, `water_and_trench1`, `pulley3`, `cat1`, `Kepler_second_law_or_area1`, `plastic_bag1`, `communicating_vessels1`, `shoes1`, and `basket1`.

Under this setting, ConceptSIR solved 9/17 tasks, covering 7 categories. Among the solved tasks, the average number of loops used until success was 2.22. Notably, candle1 and refraction2 are among the 10 tasks with 0% accuracy in Table 1, yet were solved by ConceptSIR under strict evaluation. Additionally, bead_chain1 and water_and_trench1 are tasks where only GPT-5.2 (among the four LLMs in Table 1) succeeded, while ConceptSIR also succeeded. The remaining 5 solved tasks are ones that were also solved by both GPT-5.2 and Gemini 3 Pro. Beyond solve/not-solve outcomes, we also observed that ConceptSIR improves proposal efficiency on several tasks. For example, in our LLM tests, GPT-5.2 solved bead_chain1 once within 12 proposals, whereas ConceptSIR solved it twice within 2 loops (9 proposals in total). We additionally provide the accuracy of direct LLMs and human on this subset in Table 6.

Model	Setting	Response Accuracy	Task Coverage
GPT-5.2	Language	19.61% / 19.61%	41.18% / 41.18%
	VL	21.57% / 24.51%	35.29% / 47.06%
	Overall	20.59% / 22.06%	41.18% / 47.06%
Gemini 3 Pro	Language	6.86% / 11.76%	17.65% / 29.41%
	VL	9.80% / 9.80%	23.53% / 23.53%
	Overall	8.33% / 10.78%	29.41% / 35.29%
Grok 4.1 Fast	Language	3.92% / 3.92%	11.76% / 11.76%
	VL	0.98% / 0.98%	5.88% / 5.88%
	Overall	2.45% / 2.45%	11.76% / 11.76%
Claude Opus 4.5	Language	0.00% / 0.00%	0.00% / 0.00%
	VL	0.00% / 0.00%	0.00% / 0.00%
	Overall	0.00% / 0.00%	0.00% / 0.00%
ConceptSIR	Overall	**52.94%** / –	**52.94%** / –
Human	Overall	**60.54% / 74.15%**	**100.00% / 100.00%**

Table 6: Performance on the 17-task ConceptSIR subset. Each result is reported as **Strict / +Typo**. Language uses language-only chain-of-thought prompting; VL adds an analogical prompt with visual inputs. Each direct LLM generates six responses per task in each setting and twelve responses per task overall. ConceptSIR and human results are reported only under Overall.

We acknowledge that 9/17 is not yet fully satisfactory. The limitations might be largely due to the loop budget. With more loops or more attempts, we believe ConceptSIR could solve more ARCANA tasks—in fact, in preliminary experiments with GPT-5 we observed an "evolution"-like improvement process during 15 loops on bead_chain1.

E.2 Qualitative analysis.

Here we take three tasks as examples, which are `refraction2`, `bead_chain1`, `candle1`. Restatement: `refraction2` and `candle1` are not solved by all LLM baselines, and `bead_chain1` is solved once in 12 attempts by only GPT 5.2. In ConceptSIR, `refraction2` is solved in 2 reflection steps, `candle1` is solved in 6 reflection steps. Noting that among the three codes generated by Concept execution stage, two are correct to solve task `bead_chain1` in 3 reflection steps.

As for task `refraction2` in baseline testing, GPT 5.2 generates the targets ambiguously, like "light bouncing and slow down". In ConceptSIR, Gemini 3 Pro generate a more accurate target as "Light-ray

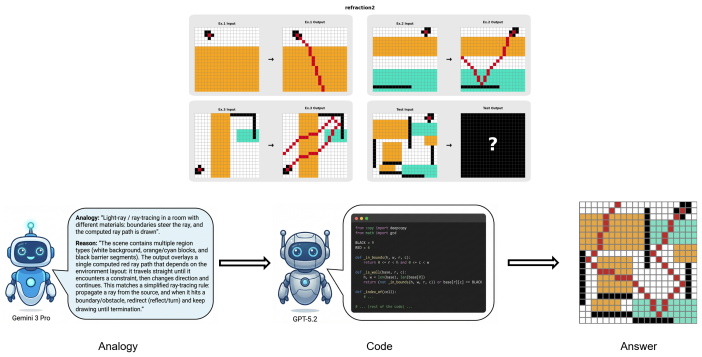


Figure 20: The reasoning trajectory of `refraction2`. Detailed concept contributes to the correct program generation and output.

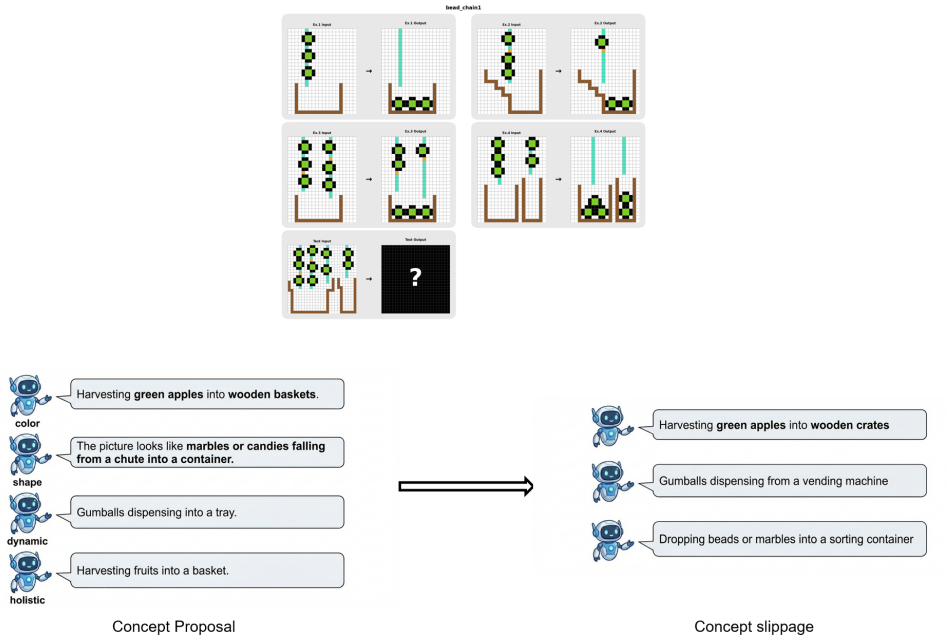


Figure 21: The concept proposal and slippage trajectory of `bead_chain1`. Concept proposal and slippage contribute better targets.

/ ray-tracing in a room with different materials", which further contributes to the correct program and output from GPT 5.2. The reasoning trajectory is shown in Figure 20.

Furthermore, we analyze the reasoning trajectory on `bead_chain1`. We note that during baseline testing, GPT 5.2 extensively uses the word “flower", e.g., “Flowers below an orange knot are replanted into the brown garden beds", “Transplanting plants into planter boxes". Noting that the targets cannot be viewed as incorrect ones, they are still different from the targets from Gemini 3 Pro (e.g., “The task is analogous to ‘beads sliding down strings’.") that are more similar to humans. In concept proposal, Gemini 3 Pro propose four different targets from color, shape, dynamics, holistic recognition, and generate three targets through concept slippage. The trajectory of concept proposal and slippage can be viewed in Figure 21.

Moreover, we analyze the reasoning trajectory on `candle1`. First, we found in step 5, GPT 5.2 generates a code that solves all of the examples, but fails on the question. The code deletes all of the candles, which is absolutely not correct. Accordingly, during the introspection, LLMs response indicates that: Missing Objects: In the "example output" generated by the code, the candles are completely missing; Missing Effects (Shadows): The code output is almost entirely yellow, meaning

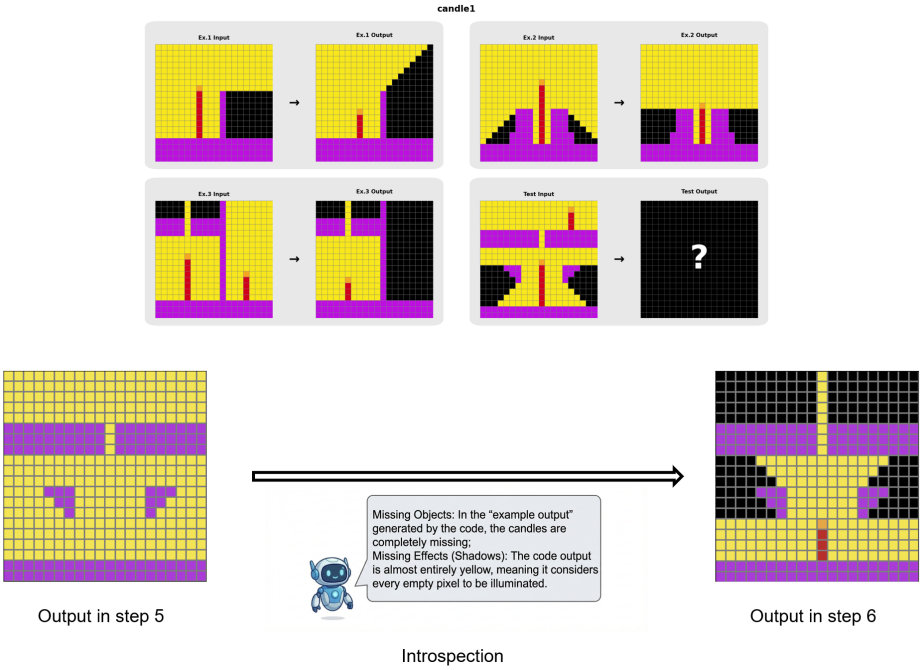


Figure 22: The introspection trajectory of `candle1`. Introspection filter invalid output.

it considers every empty pixel to be illuminated.". Introspection effectively filter the invalid output, and finally the code is refined to generate the correct answer. The introspection could be viewed in Figure 22.

During testing on ConceptSIR, we notice that the analogy accuracy of Gemini drops. We assume that during baseline testing, Gemini 3 Pro analyzes the grids in detail and notices the relations (e.g., multiple relationship) on grids, and associates it to more complicated concepts. This indicates the potential increase space for concept proposal and slippage.

F Categories and tasks visualization in ARCANA

Finally, for better understanding the diversity of ARCANA, we provide several tasks in visual. Here we list all categories and the detailed tasks in Table 7, and several tasks are shown behind.

G Further details.

Computer resources. All local scripts were run on a commodity x86_64 workstation with a 13th Gen Intel(R) Core(TM) i9-13900KF CPU (24 cores / 32 logical CPUs, up to 5.8 GHz) and 62 GiB RAM. No GPU was used, as no model training or fine-tuning is performed. Human evaluation requires no compute resources beyond data collection. LLM baseline evaluations and ConceptSIR use hosted API models; local computation consists of prompt construction, output parsing, exact-match evaluation, and lightweight Python execution for generated programs. These components are CPU-only and do not require GPUs. Wall-clock time is dominated by API latency and therefore may vary across model providers and API load.

Societal impacts. ARCANA may have positive impacts by providing a controlled benchmark for evaluating far analogical reasoning, helping diagnose limitations of current AI systems and guiding the development of more robust reasoning models. Potential negative impacts include benchmark overfitting, overinterpretation of model scores as general intelligence, and the broader dual-use risks associated with improving automated reasoning systems. We mitigate these risks by clearly stating the benchmark scope and limitations, reporting human and model results transparently, and positioning ARCANA as an evaluation resource rather than a deployment-ready capability.

Category	#	Task names
1d_wave	2	wave2, wave_interference1
2d_wave	5	Doppler1, wave1, wave_diffraction1, wave_diffraction2, wave_diffraction3
3d_cut	3	cut_sushi1, cut_sushi2, cut_sushi3
3d_move	6	brush1, mop1, mop2, pipes1, table_and_stick_and_cuboid1, table_and_stick_and_cuboid2
3d_rotate	5	book1, door1, lid1, lid2, roll_paper1
3d_shadow	5	cuboid_and_shadow1, cuboid_and_shadow2, cuboid_and_shadow3, ring1, stack1
animal	3	cat1, rabbit1, duck1
bottle	5	bottle1, bottle2, bottle3, waterstream1, waterstream2
causal_move	11	hook1, jellyfish1, toothbrush_and_toothpaste1, pen1, rocket1, shoot1,wind_and_balloon1, wind_and_balloon2, gun_revolver1, fish1
causal_updown	6	magnetic1, magnetic2, magnetic3, UFO_or_vacuum_cleaner1, sunset1, sunset2
claw	6	chopsticks1, claw1, claw2, claw3, fork1, fork2
elec_clock	2	clock4, clock5
electricity	4	car2, charge1, charge2, charge3
emoji	2	happy_and_sad1, smile1
gravity_and_liquid	4	dye1, dye2, dye3, dye4
gravity_and_solid	5	bead_chain1, car1, fall_and_dig1, plastic_bag1, portal1
heat	2	heat_transfer1, melt3
light	7	candle1, flashlight1, flashlight2, switch1, flashlight_and_stained_glass1, light1, switch_and_water_pipe1
move_and_shelter	7	basket1, cup_and_ball1, gloves1, hat1, shoes1, window1, window2
others	27	arrow1, candy_crush1, cake_and_candle1, cigar1, domino1, fire_extinguisher1, kaleidoscope1, Kepler_second_law_or_area1, lever1, lock1, logic_and_or_not1, gold_miner1, moon_and_tide1, magnetic_or_chemical_reaction1, matryoshka_doll1, nuts1, pin-hole_imaging1, golf1, puzzle_bubble1, rainbow_spring1, rust1, sandglass1, snake1, tetris1, vine1, water_and_trench1, zuma1
parabola	3	parabola1, plane_and_bomb1, portal_and_parabola1
pendulum	4	ballistic_pendulum1, bell1, newton_cradle1, newton_cradle2
plant	9	fireworks1, flower1, flower2, bomb_dropping1, tree1, tree2, tree_felling1, rain_and_umbrella1, stars1
pulley	3	pulley1, pulley2, pulley3
reflect_and_refract	5	reflection1, reflection2, refraction1, refraction2, refraction_and_reflection1
rotate_and_count	11	gear1, gear2, waterwheel1, waterwheel2, yoyo1, yoyo2, belt1, belt2, roll1, roll2, roll3
rotate_and_rotate	3	galaxy1, mechanical_arm_rotate1, pocket_watch1
rotate_and_shelter	5	disco_ball1, fan1, wind_and_flag_and_windmill1, windmill1, wind_and_flag_and_cloud1
rotate_and_tranfsform	3	hammer1, hammer2, telephone1
scissors	3	scissors1, scissors2, knife_and_cake1
shelter_then_move	6	cup1, cup2, drawer1, knife1, shelf1, toaster1
traditional_clock	3	clock1, clock2, clock3
water_deeper_concept	7	archimedes1, archimedes2, buoyancy1, buoyancy2, water_and_spoon1, water_and_spoon2, water_well1
water_level	12	communicating_vessels1, gasflow1, squeeze_sponge1, straw1, straw2, syringe1, tip1, piston1, waterflow1, waterflow2, water_dispenser1, water_dispenser2
water_transform	6	heat_water1, melt1, melt2, melt4, melt_and_frozen1, water_and_water_transformation1

Table 7: The category names and tasks in ARCANA.

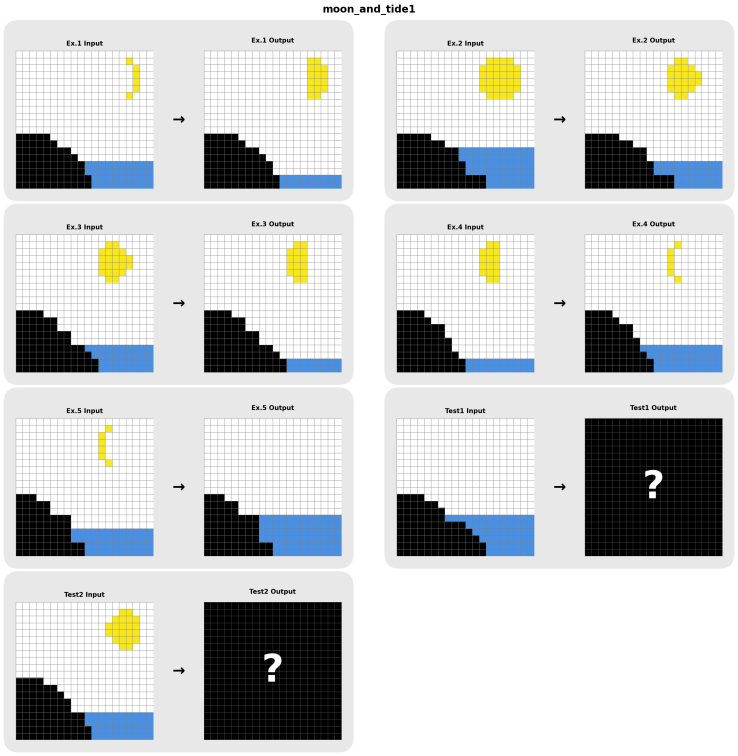


Figure 23: `moon_and_tide1` in `others` category.

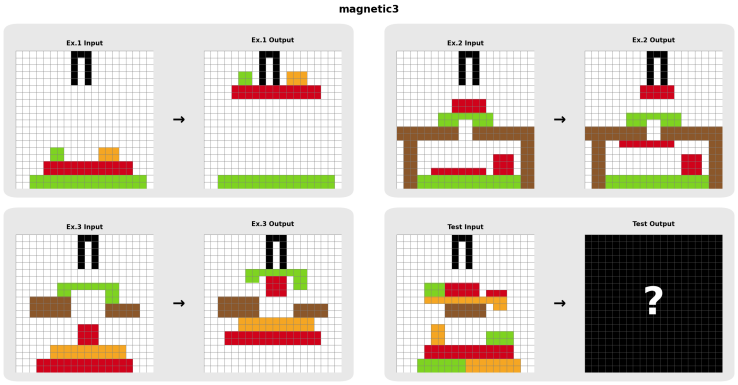


Figure 24: `magnetic3` in `causal_updown` category.

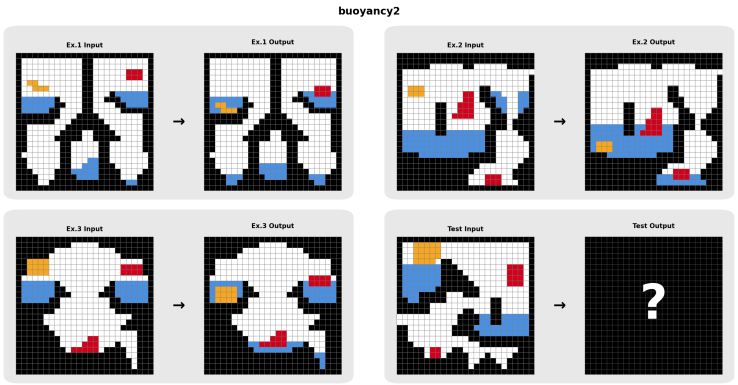


Figure 25: `buoyancy2` in `water_deeper_concept` category.

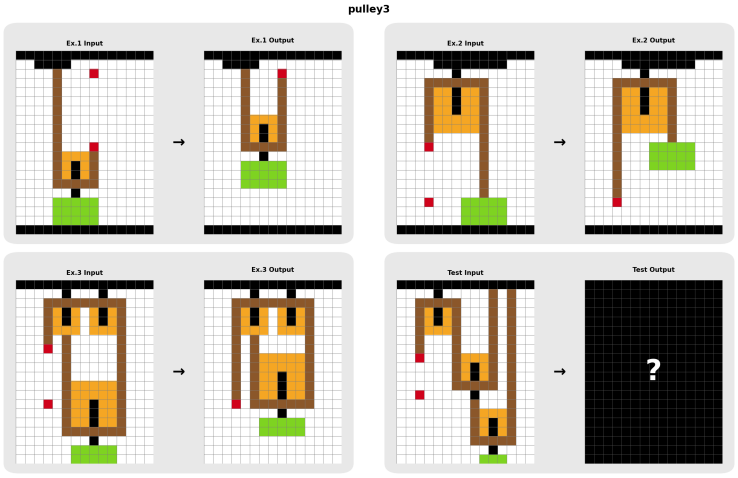


Figure 26: `pulley3` in `pulley` category.

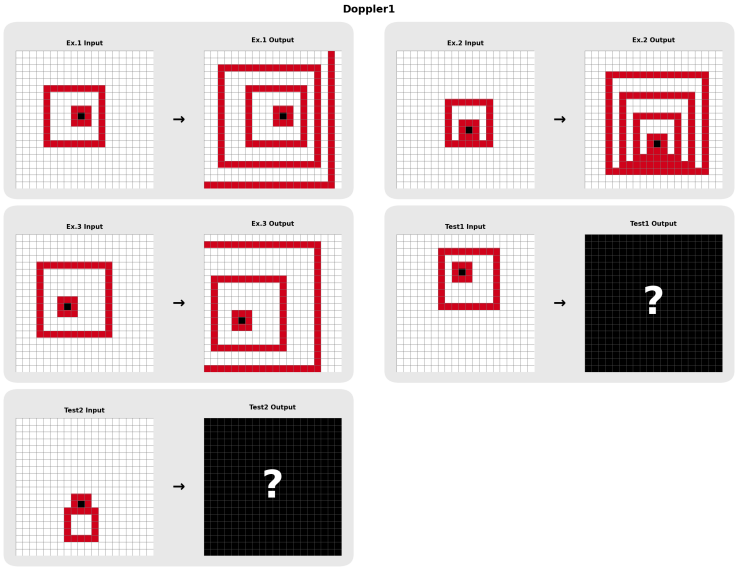


Figure 27: `Doppler1` in `2d_wave` category.

Responsible release. ARCANA is a synthetic grid-based benchmark and does not contain scraped, personal, or sensitive data. We release only task definitions, answers, prompts, and evaluation code; no pretrained model weights are released. Human-study data are anonymized or reported in aggregate, and we document the benchmark scope and limitations to reduce overinterpretation or misuse.

Licenses. ARCANA is a synthetic benchmark constructed by the authors and does not redistribute third-party proprietary data or model weights. The benchmark data are released under CC BY 4.0, and the evaluation and CONCEPTSIR code are released under the MIT License. Third-party models are accessed through hosted APIs according to their providers' terms of use.

Participant risks and ethics review. The human study consists of low-risk visual reasoning tasks. Potential risks are limited to time burden, mild fatigue, or frustration when solving difficult tasks. Participants were informed of the task format, expected participation procedure, voluntary nature of the study, and their right to stop participating. No personally identifiable information or sensitive personal data was collected, and results are reported only in aggregate. According to our institution's policy, this anonymous minimal-risk study did not require formal IRB approval.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are aligned with the paper's actual contributions: the introduction of ARCANA, the accompanying human and LLM evaluations, and the preliminary CONCEPTSIR framework. We avoid overstating the scope by presenting ARCANA as a benchmark and CONCEPTSIR as an initial method rather than a complete solution to far analogical reasoning.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of ARCANA and ConceptSIR in Section 7, and further limitations are discussed in corresponding Appendix sections.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical results or theorems requiring assumptions and proofs. The discussion of cognitive theories such as Copycat is used as conceptual motivation for CONCEPTSIR, rather than as a claimed theoretical result.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide the experimental protocols and settings that affect the paper's main claims, including dataset details, human and LLM evaluation procedures, criteria, and the configuration of CONCEPTSIR. We also provide additional details in the appendix to facilitate reproduction, though exact LLM generations may vary across API versions and stochastic decoding.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We provide the ARCANA dataset and detailed experimental protocols, but the current submission does not include the complete code needed to reproduce all main experiments. We plan to release the full code upon publication.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper does not involve training a learned model, so optimizer-related details are not applicable. We specify the relevant evaluation details, including dataset construction and splits, human-study setup, LLM model versions and prompting protocol, scoring criteria, and the CONCEPTSIR settings such as loop budget, number of runs, and stage-wise model assignment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We report uncertainty estimates for the human study using bootstrap confidence intervals and provide paired task-level bootstrap comparisons between humans and LLM baselines to quantify the human–LLM gap. For LLM baselines and the small CONCEPTSIR subset, we report exact solve counts and accuracies, and avoid claims based on LLM-to-LLM statistical significance since model ranking is not a central claim.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper does not involve training or fine-tuning any model; all LLM-based experiments use hosted API models, while local computation is limited to CPU-only prompt preparation, grid evaluation, and lightweight Python code execution in CONCEPTSIR. We specify the relevant API models, evaluation budgets and local execution environment in Appendix G; wall-clock time is mainly dominated by API latency and may vary across providers.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics: ARCANA consists of synthetic grid-based tasks without personal or sensitive data, and the human study was conducted with informed participation and without collecting personally identifiable

information. We also disclose the limitations of the benchmark and method, and do not make claims beyond the empirical evidence presented.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss that ARCANA can positively contribute to more reliable evaluation of abstract and far analogical reasoning, supporting progress in AI reasoning, cognitive science, and benchmark-driven diagnosis. We also note potential negative impacts, including benchmark overfitting, overinterpretation of LLM capabilities, and possible misuse of stronger analogical reasoning systems, and we frame ARCANA primarily as an evaluation benchmark rather than a deployed decision-making tool.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We do not release pretrained/generative model weights, scraped datasets, or data containing personal or sensitive information. ARCANA consists of synthetic grid-based reasoning tasks, and the released artifacts are limited to benchmark data, evaluation scripts, prompts, and method code, with documentation of scope and limitations.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly credit the external assets and intellectual foundations used in the paper, including ARC/ARC-AGI, Copycat-related work, and the API-based LLMs evaluated in our experiments. ARCANA consists of newly constructed synthetic tasks, and we do not redistribute third-party proprietary data or model weights; released code and data are accompanied by license and usage information.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The paper introduces ARCANA and accompanying evaluation assets, but the current submission does not yet include complete standalone documentation for all released files. We plan to provide full README files, data schemas, usage instructions, and license information with the public release.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes]

Justification: We provide the full text of the instructions given to participants, screenshots of the task interface, and details of the study protocol in the appendix. Participants were compensated at USD 6.93 per task, and the compensation scheme is reported alongside the human-study details.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [Yes]

Justification: The paper describes the minimal risks to participants, such as time burden and possible fatigue, and states that these risks were disclosed before participation. Our institution does not maintain a separately constituted IRB. Instead, the study protocol was reviewed and approved before data collection by the relevant institutional administrative office and the Academic Committee, which is formally authorized to conduct science and technology ethics review. Under our institutional governance structure, this authorized review mechanism serves the function ordinarily performed by an IRB and constitutes the institution's formal ethics determination for the study.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper explicitly describes how LLMs are used in baseline testing and in CONCEPTSIR, including model choices, stage-wise roles, prompts, evaluation protocols, and loop budgets. No LLMs are trained or fine-tuned in this work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.