Beyond One Path: Evaluating and Enhancing Divergent Thinking in Interactive LLM Agents

Anonymous ACL submission

Abstract

Divergent thinking is a core dimension of creativity, yet existing evaluations of Large Language Models (LLMs) treat them as single-turn text generations, failing to capture how an agent reasons through iterative interaction. To address this, we introduce MUTATE, an interactive benchmark designed to evaluate agentic divergent thinking at two levels: path-level, where an agent discovers multiple alternative paths to the same goal, and action-level, where individual actions require non-typical, mechanism-shifting object uses. Unlike success-only evaluations, MUTATE scores both completed paths and off-path attempts, capturing divergent reasoning that conventional success rates discard. Our experiments with frontier LLMs reveal a structural blind spot in existing frameworks: when exposed to immediate convergence pressure, they tend to fall into immediate action fixation, failing to improve action-level divergence. To overcome this, we propose ReDNA, which separates unconstrained divergent candidate generation from convergent constraint selection. ReDNA significantly outperforms prior methods across both divergence levels and generalizes effectively to an external creativity environment. We also confirm its success stems from a qualitative enhancement of resilient divergent reasoning rather than simple environmental exploration.

1 Introduction

Creativity in humans is most clearly exposed not when a problem has an obvious answer, but when the first attempt fails, and a different mechanism must be invented (Guilford, 1967; Kim, 2006). The same should be true of language model agents: their creative capacity is most informatively measured not by the polish of a one-shot answer, but by how they revise their reasoning when an interactive environment refuses the direct solution. Yet most current evaluations of Large Language Models (LLMs) still score static brainstorming outputs (Qu et al., 2025; Huang et al., 2024), leaving the interaction-driven loop of observation, action, and feedback outside the scope of measurement.

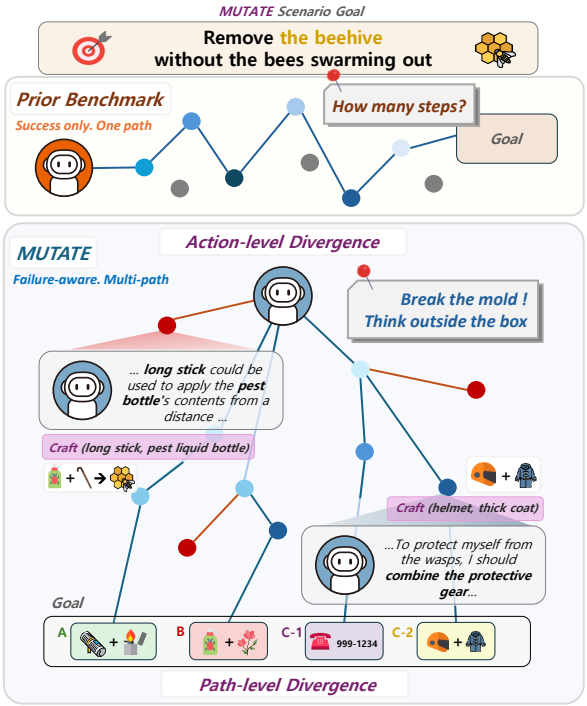


Figure 1: Example of a MUTATE scenario. Agents pursue a fixed goal in an interactive environment where multiple mechanism-distinct solution paths are valid, even failed attempts can expose divergent reasoning.

Existing simulation benchmark (Qian et al., 2024) inherits a single-path paradigm: a trajectory counts as a success only if it completes a predefined route, and every other action is discarded as noise (Figure 1, top). What is framed as evaluating divergent thinking thus reduces, in practice, to measuring convergence onto a fixed solution. Consider the beehive scenario: an agent that sprays the hive, dials an emergency number inferred from a nearby flyer, or improvises cover with available objects is exploring distinct mechanisms under environmental feedback. Yet under the single-path paradigm,

these actions enter the measurement only if they coincide with the prescribed route. Whether they succeed or fail at the goal, the divergent reasoning they exhibit at the action level is rendered invisible.

To address this, we introduce **MUTATE** (**M**ulti-path **U**nconventional **T**ask **A**ssessment with **T**ool-**E**mbedded reasoning), an interactive benchmark that separates two facets of agentic divergent thinking that conventional success rates conflate: *path-level* divergence—whether an agent finds a mechanism-distinct alternative route toward the same goal—and *action-level* divergence—whether each Thought-Action step departs from the conventional use of the object at hand. The split is deliberate: action-level breadth is a necessary but not sufficient condition for path-level discovery, so the two levels can fail independently and call for different interventions.

Figure 1(bottom) illustrates how MUTATE realizes these two levels in the beehive scenario. At the path level, the same goal admits multiple mechanism-distinct routes — wearing protection to remove the nest, calming the bees with newspaper smoke, dialing experts via a number on a nearby flyer, or luring the swarm to a garden tree — defining a discrete set of solutions against which an agent can be scored. Yet predefining a finite set of valid routes, no matter how broad, would still risk recreating the paradigm we critique: success would remain tied to completing one of the prescribed paths. MUTATE therefore complements path-level scoring with an action-level measure that scores each Thought-Action step on its own. Divergent reasoning is credited even when it does not land on any predefined route.

Benchmarking eight frontier LLMs on MUTATE reveals that agents fail at both levels through a common origin: divergent ideation and convergent selection operate under the same input conditioning, so alternatives are filtered out by the goal and prior failures before they ever enter the candidate set. To break this entanglement, we propose **ReDNA**. A Reflect module re-organizes the trajectory into a target-centered failure memory, and once failures on a target accumulate beyond a threshold, a Diverge-then-Narrowing module first proposes candidates under goal- and failure-free conditioning, then reintroduces those constraints to select among them. ReDNA improves both divergence levels on MUTATE and transfers to the Alternative Uses Test, where the same input-separation principle increases the originality and elaboration of generated ideas in a non-interactive setting.

2 Related Work

2.1 Divergent Thinking in LLMs

Divergent thinking, the ability to generate multiple novel and diverse answers to open-ended problems, has been a central indicator of creative cognition (Guilford, 1967; Kim, 2006). Recent work has quantified this ability in LLMs through standardized psychological assessments. Recent work has quantified this ability through standardized psychological assessments such as the Alternate Uses Test and the Torrance Tests of Creative Thinking, evaluating sub-dimensions like fluency, originality, and elaboration (Bellemare-Pepin et al., 2026; Zhao et al., 2025), decomposing creativity into quality, novelty, and diversity, or shifting focus to distributional diversity to expose mode collapse (Hou et al., 2025; Zhang et al., 2025a). These studies, however, evaluate output-level creativity in single-turn or segmented multi-turn settings; how divergent thinking emerges inside an interactive loop where the agent observes, acts, receives feedback, and revises remains unexplored.

2.2 Agent Evaluation in Interactive Environments

A separate line of work evaluates LLMs as agents in interactive environments, spanning text-based sandboxes (Côté et al., 2018), web agents (Zhou et al., 2024; Yao et al., 2022; Xie et al., 2024), embodied agents (Yang et al., 2025b; Zheng et al., 2024), and physical tool understanding (Zhang et al., 2025b). Because their success conditions center on goal completion, creative and divergent problem-solving largely falls outside their scope. Recent work has begun to introduce creativity into this setting: Tian et al. (2024) constructed everyday problem scenarios requiring unconventional tool use, Qian et al. (2024) extended this to interactive escape-room games with long-horizon action sequences, and multi-agent systems have explored divergent exploration, iterative refinement, and collaborative synthesis to promote creative outputs (Lu et al., 2024; Góes et al., 2023; Lin et al., 2025; Fukumura and Ito, 2025; Park et al., 2026). However, these works either do not evaluate the divergence of the step-level Thought-Action trajectory an agent produces while interacting with the environment, or do not treat the diversity of multiple solution paths discoverable for the same goal as

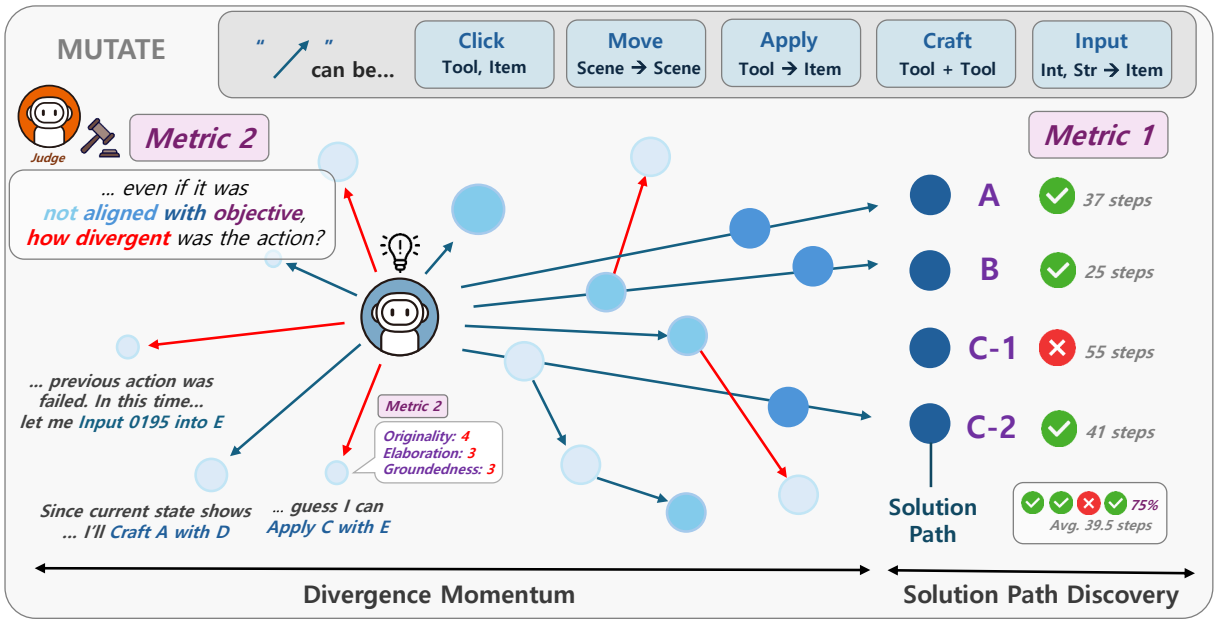


Figure 2: **Evaluation structure of MUTATE.** MUTATE separates action-level divergence, measured from Thought-Action attempts, from problem-level divergence, measured by distinct solution paths discovered for the same goal.

an object of measurement. We bridge this gap with a benchmark that simultaneously formalizes and quantifies both step-level behavioral and path-level diversity within a unified framework.

3 MUTATE

We propose **MUTATE** (**M**ulti-path **U**nconventional **T**ask **A**ssessment with **T**ool-**E**mbedded reasoning), an interactive benchmark designed to measure divergent thinking in LLM agents. As conceptualized in Figure 2, unlike conventional environments evaluating linear execution toward a single predefined solution, MUTATE places agents in text-based sandboxes where standard, intuitive approaches deliberately lead to dead-ends, forcing the agent to exhibit both action-level and path-level divergence to discover unconventional alternative paths.

3.1 Task Formulation and Metrics

Two Levels of Agentic Divergence. We formalize agentic divergent thinking into two levels, designing one distinct metric for each. *Problem-level divergence* assesses whether an agent can resiliently branch away from unexpected dead-ends to discover multiple, structurally distinct solution paths toward a single fixed goal. *Action-level divergence* evaluates whether the agent's individual Thought-Action steps, including failed attempts, break conventional object affordances through mechanism-shifting ideation. The two levels are not redundant: action-level breadth is a necessary but not sufficient condition for problem-level discovery, as a non-obvious action does not automatically constitute a goal-recoverable path.

Metric 1: Solution Path Discovery. This metric counts, across multiple trials of the same scenario, how many of the predefined valid solution paths the agent discovers. Each scenario defines four mechanism-distinct paths per phase (§3.2), so the metric directly captures problem-level divergence rather than mere repetition.

Metric 2: Divergence Momentum. This metric evaluates action-level divergence over `apply`, `craft`, and `input` attempts that did not lead to immediate success. The clearest window into an agent's divergent reasoning is the *moment* of action under environmental resistance, and even attempts that do not complete a path can reflect strong divergent thinking. We implement Metric 2 via an LLM-as-a-judge along three criteria: ***Originality*** (departure from the conventional use of the target), ***Elaboration*** (mechanistic depth of the Thought), and ***Groundedness*** (whether the Thought justifies the necessity of the Action). Originality and Elaboration follow established psychometric traditions (Guilford, 1967; Kim, 2006), while Groundedness captures the operational validity of the agentic loop. The two metrics operate jointly: completed solution paths flow into Metric 1, and every creative attempt off the solution path flows into Metric 2, leaving no part of the trajectory outside the scope

of evaluation. Details including Human-LLM correlation are provided in Appendix C.

3.2 Interactive Environment Design

Simulation Engine. MUTATE is built as a text-based, long-horizon interactive environment in the spirit of escape-room simulations: the agent navigates between scenes, manipulates items, and crafts tools to reach a fixed scenario objective, without any external hint about which mechanism the goal requires. Concretely, we build on the interactive simulation engine of Qian et al. (2024), consisting of three components: (1) *Scenes*, unit of locations connected by movement edges; (2) *Items*, interactable targets with internal states; and (3) *Tools*, collectible objects stored in the agent's *bag* that can be applied to items or crafted together. At each step, the agent observes the environment and issues one of five actions: `move(scene)`, `click(item_or_tool)`, `apply(tool,item)`, `craft(base_tool, ingredient_tool)`, or `input(string,item)`. The `craft` action advances the state of `base_tool` and is order-sensitive.

Scenarios. MUTATE consists of ten custom, manually authored, and cross-validated scenarios spanning a deliberately broad set of physical settings to prevent solutions from converging on a single domain prior. Each scenario is organized as either a single phase or two phases separated by a checkpoint, and every phase contains four mechanism-distinct solution paths classified into three categories: Affordance-based (A, directly exploiting a tool's physical property), Alternative-principle (B, achieving the same goal through a different physical or chemical mechanism), and Creative-leap (C, indirect routes such as information inference or environmental manipulation). Each phase contains one A path, one B path, and two C paths, so that problem-level divergence (Metric 1) is measured as the diversity of solution mechanisms rather than surface-level tool substitution within the same mechanism. All scene, item, and tool descriptions, as well as environmental feedback, convey only physical states and constraints and never expose the intended mechanism; otherwise, action-level divergence would collapse into hint exploitation rather than reflect genuine divergent reasoning.

3.3 Baseline Evaluation Setup

Evaluation Protocol. We evaluate the ten MUTATE scenarios by allowing the agent four independent attempts per scenario. To force path-level divergence, any solution path successfully utilized in a previous attempt is blocked and treated as unavailable in subsequent trials. A single trial terminates when the agent reaches the goal, exceeds the maximum step budget, or repeats meaningless actions for 20 consecutive steps. The length of the agent's working memory is restricted to 10 steps.

Models. For baseline calibration (§3.4), we evaluate eight models spanning four families, each with a stronger and a weaker variant: Claude Sonnet 4.6 (Anthropic, 2026) and Claude Haiku 4.5 (Anthropic, 2025), GPT-5.4 and GPT-5.4-mini (Singh et al., 2025), Qwen3-235B-A22B-2507 and Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025a), and Llama-4-Maverick and Llama-4-Scout (Meta AI, 2025). The base agent follows the ReAct (Yao et al., 2023) prompting style, producing a `Thought` and a single executable `Action` per step. All models are decoded greedily. We demonstrate the robustness of our findings through temperature setting. See Appendix D.5 for details.

3.4 Baseline Calibration and Failure Patterns

Using the setup of §3.3, we evaluate eight frontier LLMs under the base agent to verify that MUTATE measures what it claims to and to surface the structural difficulties it imposes.

Baseline Results. Table 1 reports per-model performance. Closed-source models lead in Path Discovery, with Claude Sonnet 4.6 and GPT-5.4 at the top, but even the strongest model remains well below the human average (53.6% vs. 79.9%). Crucially, the two metrics do not co-vary: Claude leads Path Discovery and Originality, while top scores on Elaboration and Groundedness belong to different models. This dispersion empirically confirms the framing of §3.1—action-level breadth is necessary but not sufficient for problem-level discovery.

Two recurring reasoning failures. Beneath the aggregate numbers, two failure patterns recur across every model regardless of family or scale. The first is *mechanism shift failure*: when one problem-solving principle is blocked, models substitute tools within the same mechanism rather than transition to a qualitatively different one. The second is *cascade construction failure*: when the cor-

Model	Metric 1: Path Discovery		Metric 2: Divergence Momentum		
	Overall	Avg Step	Originality	Elaboration	Groundedness
claude-sonnet-4.6	30 / 56 (53.6%)	26.9	**2.764**	3.079	3.350
claude-haiku-4.5	25 / 56 (44.6%)	**24.6**	2.534	**3.337**	3.320
gpt-5.4	27 / 56 (48.2%)	26.3	2.647	2.827	**3.588**
gpt-5.4-mini	9 / 56 (16.1%)	29.7	2.461	2.697	3.446
qwen3-235b-a22b-2507	22 / 56 (39.3%)	33.8	2.755	3.115	3.388
qwen3-30b-a3b-instruct-2507	7 / 56 (12.5%)	25.0	2.595	2.799	3.146
llama-4-maverick	13 / 56 (23.2%)	36.2	2.256	2.667	3.128
llama-4-scout	7 / 56 (12.5%)	36.3	2.278	2.470	2.924
Avg. human	44.75 / 56 (79.9%)	34.27	—	—	—

Table 1: Base-agent results across eight models on MUTATE. Closed-source models occupy the upper rows, open-weights models the lower rows, and the human baseline is reported as reference. Metric 2 is not computed for the human baseline. More details in Appendix. C.1

rect chain requires the tool's point of application to lie at an intermediate environmental state rather than directly on the goal, models construct the necessary tool but then apply it to the most visible target, collapsing the detour. Both failures sit at the conversion point between Divergence Momentum and Path Discovery—the moment where action-level ideation must be redirected or extended into a mechanism-distinct path. ReAct-style history, which only stacks past flat triple steps, gives the agent no structured signal about *which* target is accumulating failures, so the same mechanism is retried after repeated environmental resistance.

4 ReDNA

The diagnosis in §3.4 identifies a common origin for the two recurring failures: divergent ideation and convergent selection operate under the same input conditioning, so divergence is never free of the pressure to converge on the objective. We propose **ReDNA** (*Reflect-driven Divergent-to-Narrowing Agent*), which enforces *structural insulation* between the two operations by separating the inputs each operation is conditioned on. As shown in Figure 3, ReDNA consists of two components. The *Reflect* module re-organizes the trajectory into an object-indexed failure memory, and the *Diverge-then-Narrowing* (DN) module performs divergence and convergence as two phases whose inputs are deliberately separated.

4.1 Reflect Module

Reflect converts the raw chronological trajectory into a *target-centered* failure memory. Each entry is keyed by the target item, storing the failed action, environment response, and a repetition count incremented when the identical action–response pair recurs rather than being appended as a duplicate. The memory is updated after every step in which the agent's action targets a scene item—namely `click`, `apply`, or `input`—and the environment indicates failure. The illustration in Figure 3 (left) shows this organization: under the *beehive* target, the agent has accumulated failures from `apply(pest liquid)`, `input(911)`, and `craft(long stick, plastic bag)`, each grouped under the same target rather than scattered across history.

The DN module is invoked when the count of *substantive* failures on a single target—those produced by `apply`, `craft`, or `input` exceeds a threshold (see Appendix D.4). The point at which divergent reasoning intervenes is therefore not prespecified by a human, but is determined directly from where the agent's own trajectory has stalled.

4.2 Divergent-to-Narrowing Module

DN module temporarily replaces the policy and performs divergence and convergence as two phases.

Diverge Phase. This phase receives only the contents of the current scene, tools in the bag, and available actions. The scenario goal and the accumulated failure feedback are intentionally withheld. The model asks only "what is possible with these tools in this environment" and generates 2–4 candidate (Thought, Action) pairs. Because neither the goal nor failure feedback is present, the model avoids premature convergence—in which candidates are filtered out through self-censorship before being proposed—and can freely explore the affordance space. In Figure 3 (center), the Diverge phase proposes mechanisms such as `craft(newspaper, lighter)` to generate smoke and `apply(water hose, beehive)` to wash the hive off.

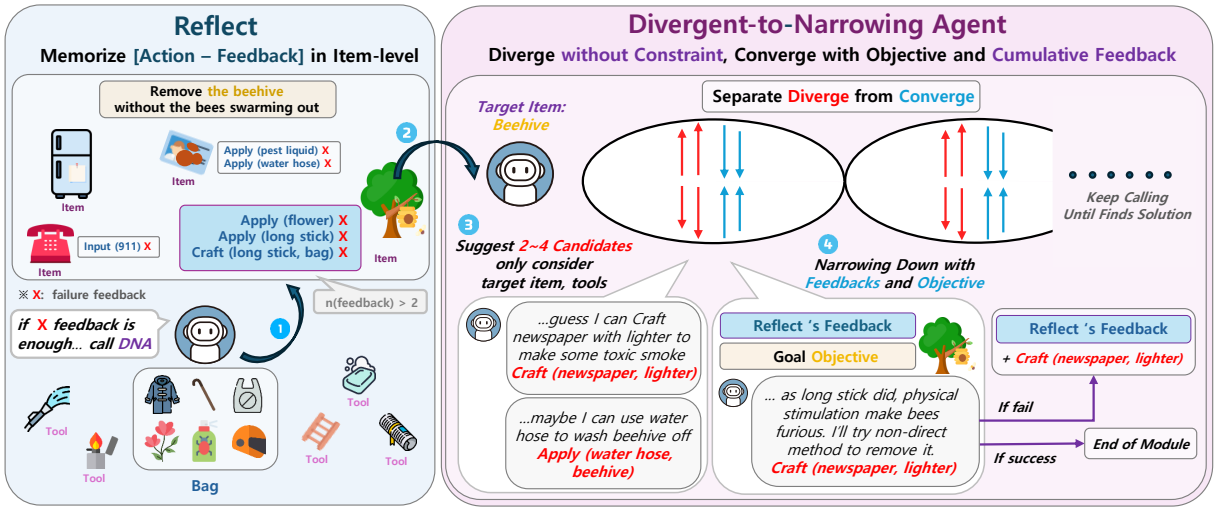


Figure 3: Architecture of ReDNA. Reflect accumulates object-level failure feedback. DN module generates candidates without convergence pressure and then selects an action based on the goal and accumulated feedback.

Narrowing Phase. This phase reintroduces the constraints that were withheld in Diverge: the candidate (Thought, Action) pairs are reunited with the scenario goal and the target-centered failure feedback that Reflect has accumulated on the current stalled item. These constraints are not external priors but the result of the agent's own interactions with the environment. The model uses them as anchors to select one of the divergent candidates. In Figure 3 (right), the water-hose candidate is filtered out—physical stimulation makes the bees more furious, consistent with the accumulated failures—and the smoke-based candidate is selected as a non-direct mechanism for the same target.

5 Experiments

5.1 Main Results

Table 2 compares Base, Self-Refine (Appendix D.2), EscapeAgent (Appendix D.3), and ReDNA across the four stronger-variant models. The interpretive frame follows from §3.4: because action-level breadth and goal-aware path discovery are distinct facets of divergence, a method that improves divergent thinking can play one of two roles—refining an already-broad ideation toward mechanism shifts, or expanding a narrow ideation space so that mechanism-distinct candidates exist in the first place. ReDNA's two phases are constructed for these two operations.

ReDNA improves Path Discovery across all models. ReDNA improves Path Discovery on all four models. The improvement holds across closed-source and open-weight models alike, indicating that the effect is not tied to a particular family or to frontier-only capability.

Prior methods broaden search but do not deepen divergence. EscapeAgent also improves Path Discovery, but its signature differs: Originality drops in every model, and Avg Step rises sharply. The added coverage comes from broader environmental exploration—more actions attempted within already-considered mechanisms—rather than from reaching qualitatively new ones. This is the mechanism shift failure of §3.4 scaled up by search rather than resolved. Self-Refine is inconsistent across models and degrades Claude Sonnet's Path Discovery, showing that a generic feasibility check is insufficient without explicit divergent–convergent structure. These results show that broadening search and deepening action-level divergence are distinct dimensions of agent behavior.

Elaboration improves universally under ReDNA. The cleanest ReDNA-specific signal appears in Elaboration: ReDNA improves it on all four models, while EscapeAgent and Self-Refine produce inconsistent or negative effects, including a substantial Elaboration drop on Claude Sonnet under EscapeAgent. ReDNA does more than expand exploration; it produces reasoning of greater mechanistic depth at each action step.

Two ideation profiles. Models split into two profiles by their base behavior (per-family analyses in Appendix B): a *high-ideation* group (Claude Sonnet 4.6, Qwen3-235B) that produces broad candidates but rotates within the same mechanism, and a *narrow-ideation* group (GPT-5.4, Llama-4-

Model	Method	Metric 1: Path Discovery		Metric 2: Divergence Momentum		
		Overall	Avg Step	Originality	Elaboration	Groundedness
GPT-5.4	Base	27 / 56 (48.2%) $_{0.00}$	**26.3** $_{0.00}$	2.647 $_{0.00}$	2.827 $_{0.00}$	3.588 $_{0.00}$
	Self-Refine	29 / 56 (51.8%) $\uparrow_{3.6}$	33.0 $\uparrow_{6.7}$	2.738 $\uparrow_{0.09}$	2.703 $\downarrow_{0.12}$	3.514 $\downarrow_{0.07}$
	EscapeAgent	35 / 56 (62.5%) $\uparrow_{14.3}$	44.9 $\uparrow_{18.6}$	2.542 $\downarrow_{0.11}$	2.990 $\uparrow_{0.16}$	3.516 $\downarrow_{0.07}$
	ReDNA	**38 / 56 (67.9%)** $\uparrow_{19.7}$	40.2 $\uparrow_{13.9}$	**2.771** $\uparrow_{0.12}$	**3.262** $\uparrow_{0.44}$	**3.725** $\uparrow_{0.14}$
Claude Sonnet 4.6	Base	30 / 56 (53.6%) $_{0.00}$	26.9 $_{0.00}$	**2.764** $_{0.00}$	3.079 $_{0.00}$	3.350 $_{0.00}$
	Self-Refine	25 / 56 (44.6%) $\downarrow_{9.0}$	**21.9** $\downarrow_{5.0}$	2.733 $\downarrow_{0.03}$	3.088 $\uparrow_{0.01}$	3.457 $\uparrow_{0.11}$
	EscapeAgent	37 / 56 (66.1%) $\uparrow_{12.5}$	43.9 $\uparrow_{17.0}$	2.526 $\downarrow_{0.24}$	2.743 $\downarrow_{0.34}$	3.341 $\downarrow_{0.01}$
	ReDNA	**41 / 56 (73.2%)** $\uparrow_{19.6}$	32.8 $\uparrow_{5.9}$	2.616 $\downarrow_{0.15}$	**3.205** $\uparrow_{0.13}$	**3.533** $\uparrow_{0.18}$
Llama 4 Maverick	Base	13 / 56 (23.2%) $_{0.00}$	36.2 $_{0.00}$	2.256 $_{0.00}$	2.667 $_{0.00}$	3.128 $_{0.00}$
	Self-Refine	13 / 56 (23.2%) $_{0.00}$	**31.2** $\downarrow_{5.0}$	2.324 $\uparrow_{0.07}$	2.681 $\uparrow_{0.01}$	**3.181** $\uparrow_{0.05}$
	EscapeAgent	23 / 56 (41.1%) $\uparrow_{17.9}$	52.7 $\uparrow_{16.5}$	2.214 $\downarrow_{0.04}$	2.571 $\downarrow_{0.10}$	3.110 $\downarrow_{0.02}$
	ReDNA	**29 / 56 (51.8%)** $\uparrow_{28.6}$	45.3 $\uparrow_{9.1}$	**2.409** $\uparrow_{0.15}$	**3.021** $\uparrow_{0.35}$	3.031 $\downarrow_{0.10}$
Qwen3 235B A22B	Base	22 / 56 (39.3%) $_{0.00}$	**33.8** $_{0.00}$	**2.755** $_{0.00}$	3.115 $_{0.00}$	3.388 $_{0.00}$
	Self-Refine	27 / 56 (48.2%) $\uparrow_{8.9}$	35.7 $\uparrow_{1.9}$	2.724 $\downarrow_{0.03}$	3.181 $\uparrow_{0.07}$	3.449 $\uparrow_{0.06}$
	EscapeAgent	34 / 56 (60.7%) $\uparrow_{21.4}$	48.3 $\uparrow_{14.5}$	2.645 $\downarrow_{0.11}$	3.148 $\uparrow_{0.03}$	**3.470** $\uparrow_{0.08}$
	ReDNA	**36 / 56 (64.3%)** $\uparrow_{25.0}$	36.5 $\uparrow_{2.7}$	2.707 $\downarrow_{0.05}$	**3.407** $\uparrow_{0.29}$	3.353 $\downarrow_{0.04}$

Table 2: Method comparison across models. Subscripts show the change relative to the same model's Base; color indicates whether the change is an improvement (green) or degradation (red), accounting for each metric's preferred direction. Bold marks the best score within each model block.

Maverick) that anchors on immediate subgoals. The two profiles differ in which side of the divergence ladder is bottlenecked.

ReDNA acts through different phases on each profile. For high-ideation models, the Narrowing phase carries weight: among the candidates the base agent already generates within a single mechanism, those exhausted by accumulated failure are excluded, and selection is redirected toward mechanism-distinct alternatives the agent's own ideation already contains but does not deploy. The Originality decrease in this group reflects this redirection of breadth into goal-aware mechanism shifts. For narrow-ideation models, the Diverge phase carries weight: candidates produced without goal or failure pressure introduce actions the base agent would not propose under standard conditioning, and Narrowing then binds them to the goal.

Case study. Figure 4 traces both ideation profiles on Zombie Lab. The two models fail in opposite ways at Base. Claude, in its high-ideation mode, inspects the gown, rag, mask, bottle, and solvent but never assembles them, so component candidates remain in the Thought line without becoming a construction. GPT, in its narrow mode, anchors on direct interactions with the crowd—extinguisher, solvent, pistol—and never proposes any delivery tool. ReDNA acts on each through a different phase.

For Claude, Reflect accumulates the rag and extinguisher failures, and Narrowing reframes them: the rag-only attempt failed for insufficient odor coverage, not because smell disguise was the wrong mechanism, so the gown is recruited to provide the missing coverage and the path completes as smell disguise (B). For GPT, the journal cue "fire cleared the door" is present in the trajectory but the Base agent reads it as license to apply the extinguisher directly; Diverge, freed from goal pressure, surfaces the missing candidate of constructing a fire-delivery tool from bottle, solvent, and cloth, and Narrowing binds it to the cue and the prior extinguisher failure, completing incendiary dispersion (C2). The two models therefore reach different mechanism-distinct paths on the same scenario, illustrating that ReDNA does not push agents toward a canonical solution but unblocks the side of the divergence ladder each profile leaves unused.

5.2 Ablation Study

ReDNA combines two components: Reflect, which accumulates object-level failure feedback, and DNA, which separates divergent candidate generation from convergent selection. To identify the source of the gains, we compare Base, an R-only ablation that retains Reflect but disables DNA, and the full ReDNA on GPT-5.4 (Table 3).

Reflect alone expands opportunity, not quality. Re-only improves Path Discovery over Base, showing that reducing repeated failures on the same ob-

Scenario: Zombie Lab. Remove the infected crowd blocking the lab entrance. Mechanism-distinct paths: A sound lure, B smell disguise, C1 vent bypass, C2 incendiary dispersion.	
Claude Sonnet 4.6 *(high-ideation)*	**GPT-5.4** *(narrow-ideation)*
Base. Inspects many objects (gown, rag, mask, bottle, solvent, cloth), but remains within single tool-on-target actions and never completes a construction.	**Base.** Anchors on direct interactions with the crowd: extinguisher, solvent, and pistol are all applied directly. It never assembles a delivery tool.
ReDNA invoked	
Reflect. `apply(rag, crowd)`: odor is insufficient. `apply(extinguisher, crowd)`: the crowd scatters only temporarily.	**Reflect.** `apply(extinguisher, crowd)`: scattering is brief and not sustained. *Journal cue:* "fire cleared the door."
Diverge. • enhance the bottle/fire route • use solvent as a flammable source • craft gown + rag as a wearable disguise	**Diverge.** • repeat direct fire-tool use • use solvent directly • construct a fire-delivery tool from bottle + solvent + cloth
Narrowing. The rag-only attempt failed due to insufficient *coverage*, not because the mechanism itself was wrong. The gown can provide coverage, while the rag provides odor.	**Narrowing.** Direct extinguisher use failed. The cue implies that fire must be *delivered*, not merely *applied*.

Action.
`craft(lab gown, infested waste rag)`
`craft(lab gown, respirator mask)`
`apply(lab gown, crowd)`
⇒ *New path: B (smell disguise)*
	Action.
	`craft(glass bottle, solvent)`
	`craft(glass bottle, cloth strip)`
	`apply(glass bottle, crowd)`
	⇒ *New path: C2 (incendiary)*

Interpretation. Narrowing-dominant: broad ideation is reorganized into a completed construction.	**Interpretation.** Diverge-dominant: a construction candidate absent from the base candidate space is newly introduced.

Figure 4: Case study of ReDNA on *Zombie Lab*. Claude benefits from Narrowing, while GPT benefits from Diverge.

Condition	Overall	Step	Orig.	Elab.	Ground.
Base	27 / 56 (48.2%)	25.3	2.67	2.73	3.28
Re only	33 / 56 (58.9%)	39.2	2.70	2.73	3.52
ReDNA	**38 / 56 (67.9%)**	36.0	**2.77**	**3.26**	**3.72**

Table 3: Ablation of ReDNA components on GPT-5.4. *R only* retains the Reflect module but disables the divergent-convergent (DNA) module.

Metric	Base	ReDNA	Δ
Originality	2.91	**3.77**	+0.86
Elaboration	4.36	**4.97**	+0.61

Table 4: Transfer to the AUT. ReDNA improves quality-oriented dimensions (Originality, Elaboration).

ject opens opportunities to reach new paths. However, action-level metrics change only marginally. Reflect, therefore, acts as an exploration enabler: it removes wasted attempts but does not change how the agent reasons within each attempt.

DNA recovers divergent ideation into target paths. Adding DNA on top of Reflect further improves Path Discovery and produces the action-level quality gains that Reflect alone does not deliver. The mechanism is the input separation of §4. Action-level ideation broadens, yet the action ultimately taken is recovered toward the target path. The contribution of ReDNA is therefore not failure memory itself—which Reflect alone already provides—but the structural separation that lets divergence operate without convergence pressure before convergence reshapes the result.

5.3 Transfer to the Alternative Uses Test

To test whether ReDNA's input-separation principle generalizes beyond interactive settings, we apply it to the Alternative Uses Test (AUT)—a single-turn psychometric task that asks for unconventional uses of a common object (Guilford, 1967). We map the principle directly: the Diverge call conditions only on the object, and the Narrowing call reintroduces the creativity criterion alongside previously generated uses, which act as the non-interactive analog of failure memory. As shown in Table 4, this yields $+0.86$ Originality and $+0.61$ Elaboration, suggesting that the divergent–convergent decoupling, not MUTATE-specific feedback, drives the gains. Details are in Appendix D.6

6 Conclusion

We introduce MUTATE, a benchmark for evaluating agentic divergent thinking via multi-path goal discovery and step-level Thought-Action divergence. MUTATE reveals two recurring failures across eight frontier models, both suggesting that agents conflate divergent generation with convergent selection. We therefore propose ReDNA, which separates candidate generation from action selection with object-level failure memory. ReDNA improves both divergence levels, generalizes to the Alternative Uses Test, and ablations show that structural separation drives the gains.

Limitations

The benchmark is based on ten human-designed scenarios and predefined gold paths, and thus needs to be validated in more diverse environments and larger-scale task sets. In addition, because step-level evaluation relies on LLM-as-a-judge, future work should strengthen evaluation reliability through larger-scale human evaluation and judge calibration. Nevertheless, this work shows that divergent thinking in LLM agents can be measured not only by success rate, but also through the joint lens of multiple path discovery and action-level ideation, and suggests that feedback-guided divergent-convergent reasoning is a promising direction for improving agentic creativity.

Ethics Statement

This work uses manually authored, simulated text environments to study divergent thinking in LLM agents. MUTATE contains no private user data or personally identifying information, and all agent interactions occur within a constrained sandbox of predefined scenes, objects, actions, and solution paths. We evaluate open-weight and API-based LLMs under documented access settings and follow the providers' usage terms. For reliability analysis, we conduct human studies in which participants solve synthetic benchmark scenarios and annotators score sampled Thought–Action records; annotators are compensated, and the results are reported only in aggregate. Because some scenarios involve fictional physical problem-solving situations, the benchmark and ReDNA are intended solely for research evaluation, not for real-world repair, emergency, or safety guidance. We also acknowledge possible subjectivity in LLM-as-a-judge scoring and therefore validate it against human annotations.

References

Anthropic. 2025. Introducing Claude Haiku 4.5. https://www.anthropic.com/news/claude-haiku-4-5.

Anthropic. 2026. Introducing Claude Sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6.

Antoine Bellemare-Pepin, François Lespinasse, Philipp Thölke, Yann Harel, Kory Mathewson, Jay A Olson, Yoshua Bengio, and Karim Jerbi. 2026. Divergent creativity in humans and large language models. *Scientific Reports*, 16(1):1279.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer.

Kazuma Fukumura and Takayuki Ito. 2025. Can llm-powered multi-agent systems augment human creativity? evidence from brainstorming tasks. In *Proceedings of the ACM Collective Intelligence Conference*, CI '25, page 20–29, New York, NY, USA. Association for Computing Machinery.

Luis Fabricio Góes, Marco Volpe, Piotr Sawicki, Marek Grses, and Jacob Watson. 2023. Pushing gpt's creativity to its limits: Alternative uses and torrance tests.

Joy P Guilford. 1967. Creativity: Yesterday, today and tomorrow. *The Journal of Creative Behavior*, 1(1):3–14.

Zhaoyi Joey Hou, Bowei Alvin Zhang, Yining Lu, Bhiman Kumar Baghel, Anneliese Brei, Ximing Lu, Meng Jiang, Faeze Brahman, Snigdha Chaturvedi, Haw-Shiuan Chang, et al. 2025. Creativityprism: A holistic benchmark for large language model creativity. *arXiv preprint arXiv:2510.20091*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *Preprint*, arXiv:2402.02716.

Kyung Hee Kim. 2006. Can we trust creativity tests? a review of the torrance tests of creative thinking (ttct). *Creativity Research Journal*, 18(1):3–14.

Yi-Cheng Lin, Kang-Chieh Chen, Zhe-Yan Li, Tzu-Heng Wu, Tzu-Hsuan Wu, Kuan-Yu Chen, Hung-yi Lee, and Yun-Nung Chen. 2025. Creativity in llm-based multi-agent systems: A survey. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 27572–27595.

Li-Chun Lu, Shou-Jen Chen, Tsung-Min Pai, Chan-Hung Yu, Hung-yi Lee, and Shao-Hua Sun. 2024. Llm discussion: Enhancing the creativity of large language models via discussion framework and role-play. *arXiv e-prints*, pages arXiv–2405.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594.

Meta AI. 2025. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Jeonghyun Park, Ingeol Baek, Seunghyun Yoon, Haeun Jang, Aparna Garimella, Akriti Jain, Nedim Lipka, and Hwanhee Lee. 2026. March: Evaluating the intersection of ambiguity interpretation and multi-hop inference. *Preprint*, arXiv:2509.22750.

Cheng Qian, Peixuan Han, Qinyu Luo, Bingxiang He, Xiusi Chen, Yuji Zhang, Hongyi Du, Jiarui Yao, Xiaocheng Yang, Denghui Zhang, et al. 2024. Escapebench: Pushing language models to think outside the box. *arXiv e-prints*, pages arXiv–2412.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Yufei Tian, Abhilasha Ravichander, Lianhui Qin, Ronan Le Bras, Raja Marjieh, Nanyun Peng, Yejin Choi, Thomas L Griffiths, and Faeze Brahman. 2024. Macgyver: Are large language models creative problem solvers? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5303–5324.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. 2025b. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *International Conference on Machine Learning*, pages 70576–70631. PMLR.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Yiming Zhang, Harshita Diddee, Susan Holm, Hanchen Liu, Xinyue Liu, Vinay Samuel, Barry Wang, and Daphne Ippolito. 2025a. Noveltybench: Evaluating language models for humanlike diversity. *arXiv e-prints*, pages arXiv–2504.

Zixin Zhang, Kanghao Chen, Xingwang Lin, Lutao Jiang, Xu Zheng, Yuanhuiyi Lyu, Litao Guo, Yinchuan Li, and Ying-Cong Chen. 2025b. Phystoolbench: Benchmarking physical tool understanding for mllms. *arXiv e-prints*, pages arXiv–2510.

Yunpu Zhao, Rui Zhang, Wenyi Li, and Ling Li. 2025. Assessing and understanding creativity in large language models. *Machine Intelligence Research*, 22(3):417–436.

Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. 2024. Towards learning a generalist model for embodied navigation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13624–13634. IEEE.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pages 15585–15606.

A Appendix Table of Contents

We organize the Appendix as follows:

1. Section B provides per-model reasoning style analyses, including representative trajectory excerpts for GPT, Claude, Llama, and Qwen families.
2. Section C validates the evaluation metrics, including LLM-as-a-judge agreement with human annotations and the human solving protocol for Metric 1.
3. Section D provides implementation details for the compared methods, including the Base agent, Self-Refine, EscapeAgent, and ReDNA.
4. Section E describes the MUTATE benchmark, including the YAML scenario structure, solution-path catalog, per-scenario path discovery, and the human-evaluation UI.
5. Section F lists the full prompt templates used for the Base agent, EscapeAgent, Self-Refine, and ReDNA.

B Per-Model Reasoning Style Analysis

Section 5.1 groups the eight models into two action-level ideation profiles—high-ideation (Claude Sonnet 4.6, Qwen3-235B) and narrow-ideation (GPT-5.4, Llama-4-Maverick)—and shows that ReDNA acts through different phases on each profile. Here we provide finer-grained reasoning style descriptions within each profile, supported by representative trajectory excerpts drawn from the base-agent runs. Table 5 summarizes the profiles; §B.1–§B.4 discuss each family with excerpts.

B.1 GPT: Goal-Aligned Local Repair

GPT models show goal-aligned, conservative, and locally coherent reasoning. The model identifies the current goal, selects a direct subgoal, chooses a grounded action, and, after failure, repairs the current hypothesis with a close variant rather than pivoting to a different mechanism.

Table 6 shows two representative excerpts. In Excerpt 1, goal alignment is high, and the model grounds its action on an explicit clue (the reversed sticky note). However, the clue–object assignment is wrong, and GPT continues to generate code variants *within the same hypothesis* rather than reassigning the clue to a different target or shifting to a non-input mechanism. This is the *cascade construction failure* of §3.4 in concrete form: the construction is locally coherent, but the mechanism is not revisited. In Excerpt 2, the model correctly extracts the "fire" clue from the journal and selects the most fire-related object available. The action is grounded, but it does not extend to constructing a *fire-delivery* object (bottle + solvent + wick), which is what the C2 path actually requires. ReDNA helps GPT primarily by expanding the candidate mechanism space in the Diverge phase before convergence selects a valid action.

B.2 Claude: High-Ideation Within-Mechanism Rotation

Claude models produce many candidate ideas in the Thought line, with detailed affordance descriptions and plausible alternatives. The surface signal is divergent, but the chosen action often remains trapped inside the same target or mechanism family.

Table 7 shows two representative excerpts. In Excerpt 1, the Thought line lists many candidates (detergent, metal pipe, coin, rolling basket, lucky number). However, every candidate orbits the same target (the rusty rear gate) and the same underlying mechanism (lubrication or leverage). This is the *mechanism shift failure* of §3.4: breadth exists at the ideation level but does not deploy as mechanism shifts. In Excerpt 2, object affordances are read with mechanistic precision (ridge pattern, fingerprint, adhesive tape). Even when a path is forbidden, the model searches for variants around the same construction. ReDNA helps Claude primarily by narrowing this already-broad ideation through accumulated failure constraints, redirecting breadth into mechanism-distinct deployment rather than within-mechanism rotation.

B.3 Llama: Exploration Without Mechanism Discipline

Llama models actively explore scenes and objects, but the update from failure feedback to a new mechanism is less reliable. The model is not simply conservative—it explores—but its exploration is weakly structured by failure evidence, so the bottleneck appears at mechanism construction rather than at exploration breadth.

Table 8 shows two excerpts from the same scenario. In Excerpt 1, scene exploration is active and the vending–change machine relation is correctly recognized. However, the model repeats actions on the same object and does not consolidate failure feedback into a precise mechanism update. In Excerpt 2, the Thought recognizes both the fire clue

Family	Style	Strengths	Limitations
GPT	Goal-aligned and conservative	Goal alignment; action grammar; physical repair sequences; low random tool use; stable groundedness.	Narrow mechanism space; delayed pivot; repeated local variants; weak multi-object construction; limited indirect delegation or symbolic transformation.
Claude	High-ideation and verbally rich	Rich elaboration; fluent candidate generation; strong affordance language; many possible tool uses; high originality.	Weak pruning; tool substitution within one target; soft use of failure feedback; delayed mechanism pivot; unstable stateful construction.
Llama	Exploratory but unstable	Active scene exploration; willingness to try alternatives; broad object probing; fast response on simple affordances; benefits from scaffolding.	Unstable feedback integration; weak Thought–Action alignment; skipped construction steps; loose objective connection; limited mechanism-level filtering.
Qwen	Mechanism-inventive but weakly disciplined	High originality; non-obvious affordance reinterpretation; mechanism construction; broad candidate space; strong benefit from failure grounding.	Weak convergence discipline; invalid tool-state assumptions; loose grammar constraints; over-wide mechanisms; unstable coverage despite originality.

Table 5: Reasoning profiles across the four model families. The high-ideation profile (Claude, Qwen) is bottlenecked by convergence to mechanism-distinct paths; the narrow-ideation profile (GPT, Llama) is bottlenecked by candidate breadth.

#	Thought	Action
Excerpt 1 — `house_heist / run_3`		
1	The reversed sticky note likely gives the safe code, so I should use that clue instead of guessing more numbers.	`input(2518, fingerprint safe)`
2	The reversed sticky note likely gives the 8-digit spare key code, so I should try that on the safe keypad.	`input(25158108, fingerprint safe)`
3	The reversed sticky note likely hides the safe code; entering a corrected 8-digit version is the strongest next step.	`input(25180000, fingerprint safe)`
Excerpt 2 — `zombie_lab / run_1`		
1	The journal strongly hints fire clears the zombies, and I have a charged extinguisher to test on the crowd as the most direct way to manipulate them.	`apply(fire extinguisher, lab entrance crowd)`

Table 6: Representative excerpts from GPT-5.4. Excerpt 1: code-variant repetition within a single hypothesis. Excerpt 2: direct application without intermediate construction.

#	Thought	Action
Excerpt 1 — `laundry_machine_7 / run_4`		
1	Rear gate is rusty and seized. Tried metal pipe, dolly, coin. Detergent is thick and slippery—could lubricate the latch, but that's forbidden. Maybe craft detergent with another tool, or try the rolling basket on the gate.	`apply(rolling basket, rear gate)`
2	Gate is rusty and seized. Detergent could lubricate the latch but forbidden. Try crafting detergent with something, or apply the coin (failed before). Let me try crafting detergent onto the metal pipe for a lubricated pipe.	`craft(metal pipe, detergent)`
Excerpt 2 — `house_heist / run_2`		
1	The wine glass shows a clear ridge pattern (fingerprint). I should use adhesive tape to lift it, then use it on the safe. But `apply(adhesive tape with fingerprint, fingerprint safe)` is forbidden. Let me try crafting adhesive tape with the fingerprint from the wine glass.	`apply(adhesive tape, wine glass)`

Table 7: Representative excerpts from Claude Sonnet 4.6. Both excerpts show many candidates surfacing in the Thought line while the chosen Action rotates within a single mechanism family.

#	Thought	Action
Excerpt 1 — `zombie_lab / run_4`		
1	I need to explore the current scene and progress towards retrieving the cure. The vending machine seems like a potential point of interest.	`click(vending machine)`
2	I need to explore and understand how to interact with the items. The vending machine and change machine seem related, and I have a bill that could be used.	`click(vending machine)`
3	The vending machine seems key to obtaining something useful, possibly by using the change machine to get coins.	`apply(bill, change machine)`
Excerpt 2 — `zombie_lab / run_4`		
1	Lab entrance is blocked by zombies. The journal hinted fire was used to clear the door. I have a fire extinguisher, which is the opposite of fire, but it might create a different solution or distract the zombies.	`click(lab entrance crowd)`
2	Lab entrance is blocked by zombies and a body jammed in the door. The journal hinted fire was used. I have a fire extinguisher, the opposite of fire, but it might be useful in a different way.	`apply(fire extinguisher, lab entrance crowd)`

Table 8: Representative excerpts from Llama-4-Maverick. Excerpt 1: action repetition despite active exploration. Excerpt 2: mechanistic insight that does not stabilize into a construction step.

and the fact that the extinguisher is the *opposite* of fire. The mechanistic insight is correct, but the action does not stabilize on the implied construction (assembling a fire-delivery object); it oscillates between re-examining the target and direct application. ReDNA helps Llama by turning scattered exploration into item-level, failure-constrained candidate selection.

B.4 Qwen: Mechanism Invention Without Grammar Discipline

Qwen frequently reinterprets objects as carriers, wedges, lures, insulators, signals, containers, triggers, or disguise components. The challenge is not the absence of ideas, but selecting which creative mechanism fits the current environment constraints and accumulated failure evidence.

Table 9 shows three excerpts spanning Qwen's characteristic patterns. In Excerpt 1, the pillow is reinterpreted as a sound dampener and the metal bookend as a mass component; the proposed mechanism is functionally coherent. Failure, however, does not redirect Qwen away from the over-wide mechanism family; it tends to extend in the same direction. In Excerpt 2, the object state is functionally reconstructed: a metal bar as a lever, grip cloth as a friction modifier. This is closer to mechanism construction than to tool substitution, and characteristic of Qwen's strength. In Excerpt 3, the odor-deception mechanism is invented by the model itself, which is characteristic of Qwen. The correct B path, however, requires a wearable disguise construction (lab gown + infected waste rag + respirator mask), not a more potent scent tool alone. Under ReDNA, Qwen's originality can be channeled toward path discovery, which can reduce measured originality in some settings because the method filters wild but weakly grounded candidates, while improving coverage by selecting task-useful mechanisms.

#	Thought	Action
Excerpt 1 — `missed_call / run_1`		
1	The music is still too loud and the small pillow alone isn't sufficient. The metal bookend is heavy and solid; combining it with the pillow might create a more effective sound-blocking tool.	`craft(small pillow, metal bookend)`
Excerpt 2 — `pipe_leak / run_3`		
1	The sealed manhole needs to be opened. The metal bar could serve as a lever, but its slick surface makes it hard to grip. The grip cloth is designed for high friction; crafting it with the metal bar would create a better tool for prying.	`craft(metal bar, grip cloth)`
Excerpt 3 — `zombie_lab / run_2`		
1	The solvent and infected waste rag may be crafted together to simulate a stronger odor for deception. Crafting them could create a more potent scent tool.	`craft(infected waste rag, solvent)`

Table 9: Representative excerpts from Qwen3-235B. Excerpts 1–2: characteristic mechanism invention. Excerpt 3: invented mechanism that does not land on the correct construction state.

Summary across families. The four families differ not only in capability level but in *which side of the divergence ladder is bottlenecked*. The high-ideation profile (Claude, Qwen) generates broad candidates but rotates within the same mechanism family; the narrow-ideation profile (GPT, Llama)

anchors on direct hypotheses or oscillates between exploration and action without consolidating mechanism construction. The asymmetric gains under ReDNA reported in Section 5.1 follow directly from this split: the Narrowing phase carries weight where ideation already exists but does not deploy, and the Diverge phase carries weight where candidates are missing in the first place.

C Evaluation Reliability

We validate the two metrics against human reference using two independent studies, one for each metric.

C.1 Metric 1: Human Solving Protocol

The human baseline reported in Table 1 (44.75/56, 79.9%) is obtained by having human participants attempt the MUTATE scenarios under the same multi-attempt protocol used for the agents (§3.3). Unlike the Metric 2 annotation, which scores existing Thought-Action records produced by models, this study has humans *solve* the scenarios themselves and counts the number of distinct solution paths they discover.

Interface. Participants interact with each scenario through the benchmark UI shown in Figure 5. The interface exposes the same observation space available to the agents—current scene description, visible items, tools in the bag, valid actions (`move`, `click`, `apply`, `craft`, `input`), and environment responses—so that human and agent attempts are evaluated under matched conditions. The Login and Tutorial views (Figure 5a-b) onboard participants; the Main and Scoring views (Figure 5c-d) display the scenario state and accept the next action; and the Context and Bag views (Figure 5e-f) surface action history and current inventory.

Protocol. We recruit four graduate students from our lab and neighboring research groups as participants. Each participant runs four independent attempts per scenario through the benchmark UI (Figure 5), which exposes the same observation space available to the agents—current scene description, visible items, tools in the bag, valid actions (`move`, `click`, `apply`, `craft`, `input`), and environment responses—so that human and agent attempts are evaluated under matched conditions. The multi-attempt protocol matches the agent setting: any solution path successfully utilized in a previous attempt is blocked and treated as unavailable in subsequent trials. Participants completed the full ten-scenario set in approximately six to seven hours each, and were compensated at 10 USD per hour. An attempt terminates when the participant reaches the goal, exceeds the maximum step budget, or repeats meaningless actions for 20 consecutive steps. Participants receive the scenario objective but no information about the underlying mechanism categories (A/B/C1/C2) or the existence of multiple paths beyond what the diversity constraint implies. We count, for each scenario, the number of distinct gold paths reached across the four attempts, and aggregate across scenarios to produce the 44.75/56 figure reported in the main text. We do not collect verbal Thought traces from participants due to the time and cost of per-action justification, so Metric 2 is not computed for the human baseline.

C.2 Metric 2: LLM-as-a-Judge and Human Annotation

Metric 2 scores non-terminal `apply`, `craft`, and `input` attempts with an LLM-as-a-judge along the three criteria. To verify that the judge tracks human reasoning, we collect parallel human scores on a sample of 296 Thought-Action records (yielding 888 criterion-level comparisons) and compare the two streams.

LLM-as-a-Judge Setup. The judge receives the scenario objective, current scene context, tools in the bag, recent history, the model's Thought, the selected Action, and the environment response. This context is necessary because the same surface action can be creative or ungrounded depending on the available tools, the current item state, and prior feedback. We use the LLM judge for large-scale scoring, and then validate its behavior against the human annotations described below, rather than treating it as a replacement for human judgment.

Human Annotation Protocol. Two graduate-level annotators fluent in English score the 296 sampled Thought-Action records under the same three criteria as the judge. Annotators are compensated at 10 USD per hour and receive written guidelines, criterion definitions, and example cases; a calibration session aligns rubric interpretation before final scoring. Each annotation item exposes the same context shown to the judge (scenario objective, environment state, action history, bag contents, Thought, selected Action, environment response), so that the two streams of scores are directly comparable. Annotators are asked to judge the attempt

Criterion	n	Pearson	Spearman	Pairwise
Overall	888	0.5564	0.5372	0.8447
Originality	296	0.4094	0.4025	0.7717
Elaboration	296	0.6227	0.5899	0.8637
Groundedness	296	0.6371	0.6193	0.8988

Table 10: Human–LLM agreement on the three Metric 2 criteria.

itself and its stated mechanism, not whether the action eventually solves the scenario.

Human–LLM Correlation Results. Table 10 reports Pearson correlation, Spearman rank correlation, and pairwise agreement between the LLM judge and human reference scores. On pairwise agreement, the LLM judge reaches 0.8447 overall, which falls within the range that Chiang and Lee (2023) identify as matching "the same level of agreement between humans" for strong LLM judges. Per-criterion pairwise agreement is consistent with this regime, with Groundedness (0.8988) and Elaboration (0.8637) at the higher end and Originality (0.7717) at the lower end. The same ordering appears in the Pearson and Spearman columns: Groundedness and Elaboration show the strongest agreement, while Originality trails, which we interpret as expected residual subjectivity, consistent with prior observations that more subjective rubric dimensions are intrinsically harder for both humans and LLM judges to agree on (Chiang and Lee, 2023). We therefore use Metric 2 primarily for aggregate comparisons and qualitative analysis, not as a claim of perfect record-level equivalence to human judgment.

Algorithm: ReDNA

Require: goal g; environment $\mathcal{E}$; base policy π_{base}; failure threshold τ; max steps T
Ensure: executed trajectory $\mathcal{T}$

Initialize failure memory $\mathcal{M} \leftarrow \emptyset$
Initialize substantive counter $c[\cdot] \leftarrow 0$
Initialize trajectory $\mathcal{T} \leftarrow \emptyset$
for $t = 1, 2, \ldots, T$ **do**
  Observe scene s_t, bag $\mathcal{B}_t$, actions $\mathcal{A}_t$
  if goal g reached **then break**
  # Policy selection based on failures
  if $\exists\, I^\star$ s.t. $c[I^\star] \geq \tau$ **then**
    # DN module: single LLM call, two phases
    # Diverge: condition on $(I^\star, s_t, \mathcal{B}_t, \mathcal{A}_t)$ *only*
    $\mathcal{C} \leftarrow \textsc{Diverge}(I^\star, s_t, \mathcal{B}_t, \mathcal{A}_t)$
      generate 2–4 candidates $\{(\text{th}_i, a_i)\}$
    # Narrowing: reintroduce $(g, \mathcal{M}[I^\star])$
    $(\text{th}^\star, a_t) \leftarrow \textsc{Narrow}(\mathcal{C}, g, \mathcal{M}[I^\star])$
  else
    $(\text{th}^\star, a_t) \leftarrow \pi_{\text{base}}(s_t, \mathcal{B}_t, \mathcal{A}_t, g)$
  end if
  Execute a_t in $\mathcal{E}$; receive response r_t
  Append $(\text{th}^\star, a_t, r_t)$ to $\mathcal{T}$
  // Reflect: target-centered memory update
  if a_t targets item I **and** r_t is failure **then**
    if $a_t \in \{\texttt{click}, \texttt{apply}, \texttt{input}\}$ **then**
      if $(a_t, r_t) \in \mathcal{M}[I]$ **then**
        increment repetition count
      else $\mathcal{M}[I] \leftarrow \mathcal{M}[I] \cup \{(a_t, r_t)\}$
      end if
    end if
    if $a_t \in \{\texttt{apply}, \texttt{craft}, \texttt{input}\}$ **then**
      $c[I] \leftarrow c[I] + 1$ // substantive
    end if
  end if
end for
return $\mathcal{T}$

Table 11: ReDNA algorithm.

D Method Implementation Details

Table 12 summarizes the implementation differences among methods. We then describe each method in the same order.

D.1 Base Agent

The Base agent (Yao et al., 2023) is the standard action policy used for comparison. We implement ReAct in a zero-shot setting, where environment responses are recorded in the recent interaction history without explicit Observation labels. At each step, we provide the current scene description, possible actions, tools in the bag, recent interaction history, and the diversity constraint that prevents reuse of already discovered finish actions. The LLM outputs exactly one next action in the two-line `Thought:`/`Action:` format, following the Thought–Action structure of ReAct.

The Base agent does not maintain a task list or structured failure memory. Failed actions remain in the recent history, but we do not separately abstract which target item accumulates failures or which mechanism repeatedly fails. Its decision process is therefore local to the current scene and recent trajectory.

D.2 Self-Refine Agent

The Self-Refine (Madaan et al., 2023) agent adds a feasibility check to the Base prompt. We ask the model to verify, before writing the final action, whether the action is physically plausible, follows the action grammar, uses tools already in the bag,

Method	State	Intervention	Role
Base	Recent history only	Single action policy	Provides the standard local decision baseline.
Self-Refine	Recent history only	Feasibility self-check	Reduces invalid or weakly grounded actions.
EscapeAgent	Task list and forethought queue	Reflection plus candidate generation	Expands exploration by proposing new tool-task candidates.
ReDNA	Target-centered failure memory	Divergent generation followed by constrained selection	Uses failure evidence to select mechanism-level alternatives.

Table 12: Method summary.

targets a current-scene item for `apply`, uses two bag tools for `craft`, uses a valid movement label for `move`, and avoids clearly repeated failures.

Self-Refine does not add external memory or new exploration state. Its role is to reduce invalid or weakly grounded actions rather than to expand the mechanism space.

D.3 EscapeAgent

EscapeAgent (Qian et al., 2024) adds task-list reflection and forethought modules on top of the Base agent. After an action and environment response, the reflection module updates a task list with unresolved targets and feedback. When the agent collects a new tool or discovers a new task, a forethought prompt proposes possible `craft`, `apply`, `click`, or `input` candidates.

This design broadens candidate generation. However, the generated candidates are not always constrained by item-level failure evidence. EscapeAgent can therefore produce many superficially creative tool-target proposals whose functional relation to the current obstacle is weak. In *Zombie Lab*, for example, EscapeAgent can discover all paths, but it also spends actions on proposals such as applying currency or protective equipment to unrelated shelves.

D.4 ReDNA

ReDNA combines target-centered failure memory with a feedback-guided divergent-convergent reasoning module, formalized in Algorithm 11. We retain the Base policy for ordinary steps, and once the substantive failure counter $c[I^\star]$ on some target item crosses the threshold τ, the DN module temporarily takes over. Table 13 summarizes the full per-step procedure, and Table 14 lists the hyperparameter values used in our experiments.

The reflection memory $\mathcal{M}$ stores failed interactions by the target item. For a target such as `lab entrance crowd`, failures like `apply(infected waste rag, lab entrance crowd)` and `apply(fire extinguisher, lab entrance crowd)` become accumulated evidence about the same obstacle, with identical (a_t, r_t) pairs merged via a repetition count rather than appended as duplicates. The counter $c[I]$ is incremented only for `apply`, `craft`, and `input`, excluding `click` since it primarily serves observation, so DN is triggered by mechanism-level resistance rather than by inspection failures. We use this memory not only to avoid exact repetition, but also to diagnose why a mechanism fails and what missing property a new mechanism should provide.

The DN module has two phases. In the divergent phase, the model generates two to four candidate mechanisms from the current target item $I^\star$, scene, and bag tools, with the scenario goal g and the failure memory $\mathcal{M}[I^\star]$ intentionally withheld to prevent premature convergence. In the convergent phase, the model reunites these candidates with the objective, possible actions, diversity constraint, and the accumulated failure memory $\mathcal{M}[I^\star]$, and selects one valid `apply`, `input`, or `craft` action. This lets ReDNA expand alternatives near the failed target while still selecting a grounded action.

D.5 Robustness to Decoding Stochasticity

The main results in Table 1 use greedy decoding. To verify that the qualitative claims of §5.1 are not artifacts of a single deterministic pass, we re-run every (model, method) cell on the four stronger-variant models at temperature $T = 0.7$, three independent runs each. All other components of the protocol—scenarios, four-attempt evaluation, forbidden-finish-action diversity constraint, step budget, working-memory cap, and the LLM-as-a-judge—are kept identical to §3.3.

Table 15 reports mean $\pm$ sample standard deviation across the three runs. The two main-text directions hold under stochastic decoding: ReDNA achieves the highest Path Discovery on all four

Stage	Operation
(1) Per-step memory update (Reflect)	
Trigger	every step in which the action targets a scene item
Record	store (a, r) under the target item if $a \in \{\texttt{click}, \texttt{apply}, \texttt{input}\}$ and r is a failure; merge identical (a, r) pairs by incrementing a count rather than appending duplicates
Substantive counter	increment a separate per-target count when $a \in \{\texttt{apply}, \texttt{craft}, \texttt{input}\}$ (excluding `click`)
(2) DN invocation check	
Condition	DN module is invoked at the next step iff the substantive counter on some target T reaches threshold τ
Otherwise	the base ReAct policy decides the next action
(3) Diverge phase (candidate generation)	
Inputs supplied	target T; scene context; tools in bag; available actions
Inputs withheld	scenario goal g; accumulated failure memory $\mathcal{M}[T]$
Output	2–4 candidate (Thought, Action) pairs $\{(t_i, a_i)\}$, generated under goal- and failure-free conditioning
(4) Narrowing phase (constraint-anchored selection)	
Inputs supplied	candidate set $\{(t_i, a_i)\}$; scenario goal g; failure memory $\mathcal{M}[T]$
Selection rule	choose one $(t^\star, a^\star)$ that is consistent with g and not contradicted by entries in $\mathcal{M}[T]$
Output	a single executable action $a^\star$ to be issued to the environment

Table 13: Procedure of ReDNA per environment step.

models, with gaps to the next-best method that exceed one standard deviation in every case; and ReDNA produces the highest Elaboration on all four models. Run-to-run variance is small in absolute terms—under 6 percentage points on Path Discovery and under 0.10 on action-level metrics in every cell—so the conclusions in §5.1 that rest on gaps larger than this scale are not driven by stochasticity.

D.6 AUT Implementation Details

We evaluate both Base and ReDNA on the AUT using GPT-5.4 with sampling temperature $T = 0.7$, averaged over three independent runs per condition. Base produces all uses in a single call. ReDNA follows the input-separation structure of §4: the Diverge call generates candidate uses conditioned only on the object, and the Narrowing call selects and elaborates them under the creativity criterion, using previously generated uses from earlier runs as the non-interactive analog of failure memory. Originality and Elaboration are scored on a 1–5 scale following the rubric of (Guilford, 1967).

Setting	Value
Tracked actions	`apply`, `craft`, `input`
Excluded actions	`move`, `click`
Memory key	Target item
Trigger condition	Repeated same-target failure with non-click evidence
Candidate count	2–4 mechanisms
DN final actions	`apply`, `input`, or `craft` only
Diversity control	Previously discovered finish actions are forbidden
Inference temperature	$T = 0$

Table 14: ReDNA settings.

Use of Large Language Models. We write the manuscript ourselves. We use an LLM (ChatGPT) solely for language refinement, including style, clarity, and grammar, and do not use it for research ideation or analysis.

E MUTATE Benchmark Details

E.1 YAML Scenario Structure

We store each benchmark instance as a YAML file. A file represents one interactive scenario and is encoded as a list of scenes. The first scene also contains the scenario-level `objective`, which we show to the agent as its goal. Listing 1 shows the schema skeleton.

Listing 1: YAML scenario skeleton.

```
- name: scene_name
  objective: "Scenario goal."   # first scene
   only
  desc: "Textual scene description."
  visible: true                 # optional;
  default true
  scene_relations:
    Move label: other_scene_name

  items:
  - position: "Item location."
    item:
      name: item_name
      visible: true             # optional;
  default true
      interactable: true        # optional;
  default true
      states:
      - desc: "Item state description."
        neg_reward: "Failure feedback."
        transitions:
        - wait_for: [apply, tool_name]
          trigger: [change_state, 1]
          reward: "Success feedback."
```

Model	Method	Path %	Avg Step	Originality	Elaboration	Groundedness
GPT-5.4	Base	47.0 ± 5.8	29.8 ± 1.3	2.775 ± 0.021	2.829 ± 0.030	3.519 ± 0.046
	Self-Refine	45.8 ± 2.7	31.0 ± 4.6	2.765 ± 0.054	2.737 ± 0.011	3.490 ± 0.064
	EscapeAgent	61.3 ± 2.7	38.7 ± 1.6	2.441 ± 0.024	2.919 ± 0.067	3.573 ± 0.012
	ReDNA	64.9 ± 2.7	40.8 ± 2.6	2.713 ± 0.048	3.304 ± 0.030	3.715 ± 0.069
Claude Sonnet 4.6	Base	50.0 ± 3.1	29.0 ± 1.4	2.664 ± 0.003	3.051 ± 0.033	3.346 ± 0.037
	Self-Refine	48.8 ± 5.8	21.3 ± 1.1	2.739 ± 0.020	3.072 ± 0.019	3.377 ± 0.098
	EscapeAgent	55.9 ± 3.7	37.9 ± 3.0	2.475 ± 0.031	2.813 ± 0.006	3.323 ± 0.048
	ReDNA	61.3 ± 5.4	28.9 ± 1.0	2.618 ± 0.017	3.092 ± 0.029	3.461 ± 0.045
Llama-4-Maverick	Base	24.4 ± 3.7	28.8 ± 2.1	2.142 ± 0.088	2.662 ± 0.057	3.189 ± 0.031
	Self-Refine	23.8 ± 1.0	33.4 ± 0.9	2.301 ± 0.061	2.713 ± 0.015	3.137 ± 0.061
	EscapeAgent	29.8 ± 2.7	43.3 ± 3.3	2.213 ± 0.058	2.628 ± 0.057	3.159 ± 0.018
	ReDNA	53.0 ± 2.7	45.1 ± 2.5	2.330 ± 0.008	3.074 ± 0.061	3.088 ± 0.042
Qwen3-235B-A22B	Base	40.5 ± 2.7	32.4 ± 4.5	2.738 ± 0.077	3.228 ± 0.072	3.431 ± 0.013
	Self-Refine	41.7 ± 2.1	33.3 ± 0.2	2.696 ± 0.068	3.203 ± 0.055	3.459 ± 0.032
	EscapeAgent	48.2 ± 3.1	40.2 ± 2.7	2.695 ± 0.044	3.197 ± 0.050	3.459 ± 0.039
	ReDNA	56.0 ± 1.0	35.8 ± 2.6	2.692 ± 0.055	3.424 ± 0.055	3.375 ± 0.045

Table 15: Robustness study at T=0.7, three runs per cell; values are mean $\pm$ sample standard deviation.

```
tools:
- position: "Tool location."
  tool:
    name: tool_name
    visible: true
    states:
    - desc: "Tool description."
      apply_to: [target_item_name]
      wait_for: [ingredient_tool_name]
```

We treat a scenario as a stateful text environment. The agent occupies one scene at a time and observes the current scene description, visible items, visible tools, nearby scenes, and tools already collected in its bag. The `scene_relations` field defines the navigation graph: each key is the movement label exposed to the agent, and each value is the destination scene.

We divide objects into items and tools. Items are scene objects that can be inspected or manipulated but are not normally collected. Tools are collectable objects; when the agent executes `click(tool)`, the tool moves into the bag and can later be used in `apply(tool, item)` or `craft(tool_a, tool_b)` actions.

Each item or tool may have multiple states. The current state determines its visible description and available interactions. Item states may define transitions, where `wait_for` specifies the action pattern, `trigger` specifies the world-state update, and `reward` gives the environment feedback. If no transition matches, the environment returns `neg_reward` when provided, or a generic failure message otherwise.

The action space is fixed: `click(x)` inspects an item or collects a visible tool; `apply(t, x)` applies a collected tool to an item; `craft(t1, t2)` combines two collected tools; `input(s, x)` enters a string into an item; and `move(scene)` moves to an adjacent scene.

The `trigger` field supports state changes such as changing the current item state, changing another item or tool state, toggling visibility, toggling interactability, or converting an item into a collectable tool. Successful terminal solutions are marked textually in the transition reward with `GAME END!`. Some two-phase scenarios contain intermediate rewards marked with `CHECKPOINT!`; we use them to separate Phase 1 and Phase 2 solution discovery. Separate evaluation metadata files specify valid solution path IDs, trigger actions, phases, and principal types.

E.2 Solution Path Catalog

Tables 16 and 17 list the gold path catalog. Each phase contains one A path, one B path, and two C paths unless the scenario intentionally leaves a path unavailable.

E.3 Path Discovery per Scenario

We report per-path discovery with the model-method entries for which per-path coverage is available. Table 18 reports two-phase scenarios, and Table 19 reports single-phase scenarios. O indicates that the path is discovered, and X indicates that it is not discovered.

Scenario	Phase	A	B	C1	C2
House Heist	Safe Access	Break the safe with the pipe head.	Recover a wine-glass fingerprint with adhesive tape.	Cut the power cable.	Decode binary override `10010110`.
	Room Escape	Use the door key.	Break the window pane with scissors.	Use the curtain hook and chimney opening.	Decode reversed door code `2518`.
Desert Van Restart	Tire Mobility	Replace the wheel.	Patch and reinflate the tire.	No tire-side C1 path.	Call desert recovery service.
	Engine Restart	Replace the hose.	Seal the hose with gum or tape.	Use a donor engine part.	Call desert recovery service.
Laundry Machine 7	Basement Access	Move machine 2 with the dolly.	Open the vent grate with a coin.	Loosen the rusted handle with detergent.	Infer passcode `0411`.
	Machine Stop	Input timer code `000`.	Damage the drain hose with the fire axe.	Map digit-7 strokes to `abceg`.	Select service option `3`.
Sinking Ship	Deck Access	Infer door code `1704`.	Open the maintenance hatch.	Use the duct route.	Restore the service lift.
	Rescue/Escape	Fire the flare gun.	Repair and inflate the lifeboat.	Restore radio channel `16`.	Operate winch/davit release.

Table 16: Two-phase solution paths.

Scenario	A	B	C1	C2
Beehive Removal	Wear protection and remove the nest.	Calm bees with newspaper smoke.	Restore the flyer number and call experts.	Use floral lure toward the garden tree.
Missed Call	Cover the speaker with a comforter.	Stop the laptop with the bookend.	Infer the laptop password.	Close the closet door.
Pipe Leak	Tighten the pipe joint.	Seal the pipe with a rubber patch.	Input pressure code `1225`.	Open the manhole to drain water.
Space Station Fire	Open bulkhead with code `1969`.	Use extinguisher propulsion.	Use the pressure dump lever.	Use the zero-g crawlway.
Underground Mine	Use sedative jerky.	Repel the dog with sulfur odor.	Input voice keyword `blaze`.	Burst the ceiling pipe with explosives.
Zombie Lab	Use the cola noise lure.	Build an infected scent disguise.	Open the vent route.	Craft a firebomb.

Table 17: Single-phase solution paths.

E.4 Benchmark UI for Human Evaluation

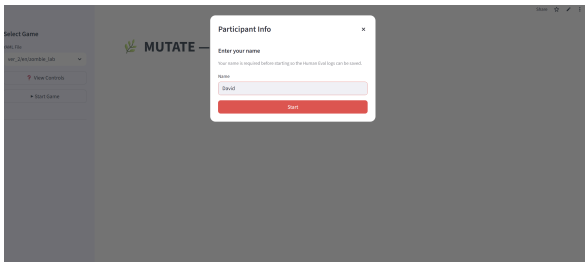
We implement the human evaluation interface as a web UI. Annotators log in, read the tutorial, then score sampled Thought-Action records against the displayed environment state, action history, current bag contents, and scenario objective. Figure 5 shows the login, tutorial, scoring, context, and bag views.

Scenario	Method	P1_A	P1_B	P1_C1	P1_C2	P2_A	P2_B	P2_C1	P2_C2
House Heist	Claude Base	X	O	O	X	X	O	X	O
House Heist	Claude ReDNA	X	O	O	X	O	O	X	X
House Heist	GPT Base	X	O	O	X	X	O	O	X
House Heist	GPT ReDNA	O	X	O	X	X	O	O	X
House Heist	Llama Base	X	O	X	X	O	X	X	X
House Heist	Llama ReDNA	X	X	O	X	X	X	X	X
House Heist	Qwen Base	X	X	X	X	X	X	X	X
House Heist	Qwen ReDNA	X	O	O	X	O	X	O	X
Desert Van	Claude Base	O	O	X	X	O	O	O	X
Desert Van	Claude ReDNA	O	O	X	X	O	O	O	X
Desert Van	GPT Base	O	O	X	X	O	X	O	X
Desert Van	GPT ReDNA	O	O	X	X	O	O	O	X
Desert Van	Llama Base	X	X	X	X	X	X	X	X
Desert Van	Llama ReDNA	O	X	X	X	O	O	X	X
Desert Van	Qwen Base	X	O	X	X	X	X	X	X
Desert Van	Qwen ReDNA	O	O	X	X	O	O	X	X
Laundry 7	Claude Base	X	O	O	X	X	X	O	X
Laundry 7	Claude ReDNA	X	O	O	O	O	X	O	X
Laundry 7	GPT Base	O	O	X	O	X	X	X	X
Laundry 7	GPT ReDNA	X	O	O	X	X	O	X	O
Laundry 7	Llama Base	X	O	X	X	O	X	X	X
Laundry 7	Llama ReDNA	O	O	O	X	O	X	X	X
Laundry 7	Qwen Base	X	O	X	O	X	X	X	X
Laundry 7	Qwen ReDNA	O	O	O	O	O	X	X	O
Sinking Ship	Claude Base	X	O	O	O	X	O	X	X
Sinking Ship	Claude ReDNA	X	O	O	O	X	O	O	O
Sinking Ship	GPT Base	X	O	O	O	X	O	X	O
Sinking Ship	GPT ReDNA	X	O	O	O	X	O	O	O
Sinking Ship	Llama Base	X	O	X	X	X	X	X	X
Sinking Ship	Llama ReDNA	X	O	O	O	X	O	O	O
Sinking Ship	Qwen Base	X	O	O	O	X	O	O	X
Sinking Ship	Qwen ReDNA	O	O	O	O	X	O	O	O

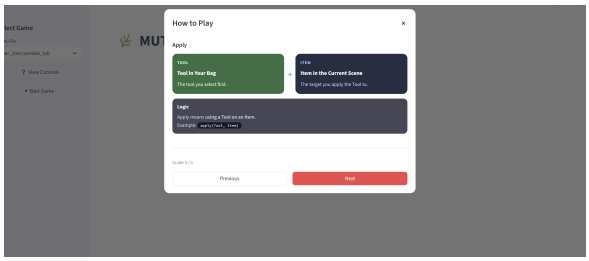
Table 18: Two-phase path discovery.

Scenario	Method	A	B	C1	C2
Beehive	Claude Base	O	O	X	O
Beehive	Claude ReDNA	O	O	X	O
Beehive	GPT Base	O	X	X	X
Beehive	GPT ReDNA	O	O	O	O
Beehive	Llama Base	O	X	X	X
Beehive	Llama ReDNA	O	O	X	X
Beehive	Qwen Base	O	X	O	X
Beehive	Qwen ReDNA	O	X	O	X
Missed Call	Claude Base	X	O	X	O
Missed Call	Claude ReDNA	O	O	X	O
Missed Call	GPT Base	X	X	X	O
Missed Call	GPT ReDNA	X	X	X	O
Missed Call	Llama Base	X	X	X	X
Missed Call	Llama ReDNA	O	X	X	O
Missed Call	Qwen Base	X	X	X	O
Missed Call	Qwen ReDNA	O	X	X	O
Pipe Leak	Claude Base	O	X	X	X
Pipe Leak	Claude ReDNA	O	O	O	O
Pipe Leak	GPT Base	O	O	O	X
Pipe Leak	GPT ReDNA	O	O	O	O
Pipe Leak	Llama Base	O	O	X	O
Pipe Leak	Llama ReDNA	O	O	X	O
Pipe Leak	Qwen Base	O	O	O	O
Pipe Leak	Qwen ReDNA	O	O	O	X
Space Fire	Claude Base	O	X	O	O
Space Fire	Claude ReDNA	O	O	O	O
Space Fire	GPT Base	X	O	O	O
Space Fire	GPT ReDNA	O	O	O	X
Space Fire	Llama Base	X	O	O	O
Space Fire	Llama ReDNA	O	O	O	O
Space Fire	Qwen Base	O	X	O	O
Space Fire	Qwen ReDNA	O	X	O	O
Mine	Claude Base	O	X	O	X
Mine	Claude ReDNA	O	X	O	X
Mine	GPT Base	O	X	O	X
Mine	GPT ReDNA	O	X	O	O
Mine	Llama Base	O	X	X	X
Mine	Llama ReDNA	O	X	X	X
Mine	Qwen Base	O	X	X	X
Mine	Qwen ReDNA	O	X	O	X
Zombie Lab	Claude Base	O	X	O	X
Zombie Lab	Claude ReDNA	O	O	O	O
Zombie Lab	GPT Base	O	X	X	X
Zombie Lab	GPT ReDNA	O	X	O	O
Zombie Lab	Llama Base	X	X	X	X
Zombie Lab	Llama ReDNA	O	X	O	X
Zombie Lab	Qwen Base	O	X	O	O
Zombie Lab	Qwen ReDNA	X	O	O	O

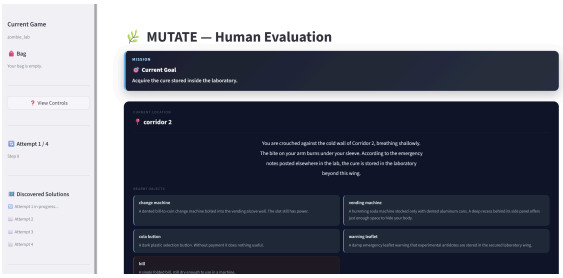
Table 19: Single-phase path discovery.



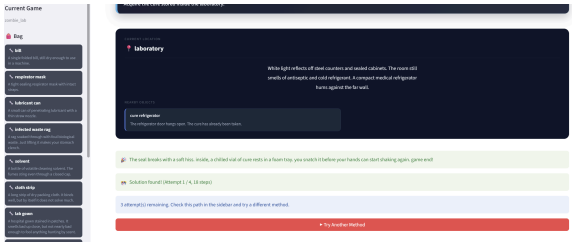
(a) Login.



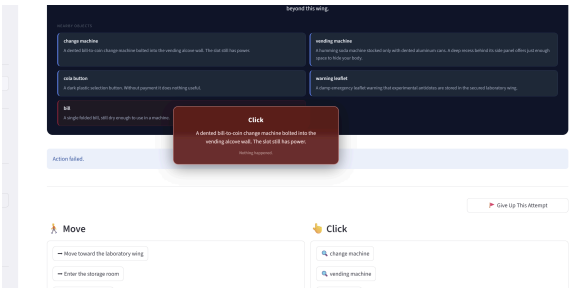
(b) Tutorial.



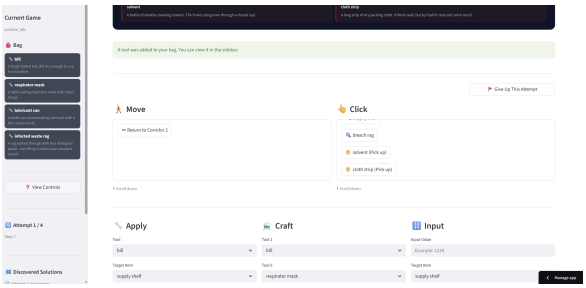
(c) Main view.



(d) Scoring view.



(e) Context view.



(f) Bag view.

Figure 5: Human evaluation UI.

F Prompts

This section lists the full prompt templates used for the Base agent, EscapeAgent, Self-Refine, and ReDNA.

```
You are interacting with a simulated environment. Your objective is to accomplish the scenario's functional
      goal (e.g., stabilize a system, repair something, restore operation).
IMPORTANT: There may be MULTIPLE valid ways to achieve the goal. Try different approaches and discover
      alternative solution paths when possible.

There are three types of interactives:
- items: interactable objects in the CURRENT SCENE
- tools: usable tools already collected IN YOUR BAG
- scenes: nearby scenes you can move to

You can perform ONE of the following actions:
- click(<interactable item or visible tool>): Examine an item in the current scene, or collect a visible tool
      from the current scene into your bag.
- apply(<tool in your bag>, <interactable item in the current scene>): Apply a tool in your bag to an ITEM in
      the current scene.
- input(string, <interactable item in the current scene>): Input a string to an item in the current scene.
- move(<interactable scene>): Move to a nearby scene.
- craft(<base tool in your bag>, <ingredient tool in your bag>): Apply the ingredient tool onto the base tool
      to upgrade or modify the base tool. The FIRST argument is the base tool. The SECOND argument is the
      ingredient applied onto it. Order matters: craft(a, b) is NOT craft(b, a).

Examples:
click(microwave)
click(wrench)
apply(key, silver chest)
craft(controller, battery)
input(c79a1, combination lock)
move(Go to operation room)

CRITICAL OPERATIONAL RULES:
1) You may ONLY use names explicitly listed in Possible Actions.
2) A visible tool in the scene is NOT in your bag yet. If you want to use it later, click it first.
3) craft can ONLY be used between TWO TOOLS already in your bag. The FIRST argument is the base tool. The
      SECOND argument is the ingredient tool.
4) apply can ONLY target an INTERACTABLE ITEM IN THE CURRENT SCENE. The second argument of apply() must appear
      under Possible Actions items. If the target is a tool, use craft().
5) If you are considering craft or apply, first check whether both required tools are already in your bag, and
      whether the target is an item rather than a tool.
6) If an action fails, do not repeat the same failed hypothesis immediately. Change the target, collect missing
       tools, move, or try a different mechanism.
7) Prefer actions that increase future options: explore scenes, click unexplored items, collect relevant tools,
       then apply or craft from observed evidence.
8) Because there can be multiple solutions, do not assume there is only one correct next step.
9) After craft(A, B) SUCCEEDS, do NOT attempt craft(A, B) or craft(B, A) again. The ingredient B is consumed on
       success.

ANTI-LOOP RULES:
- If an action fails ONCE, do NOT repeat the exact same action.
- NEVER use apply(X, Y) where Y is a tool in your bag or a visible tool in the scene. Use craft() for tool-on-
      tool operations.
- After craft(A, B) succeeds, never repeat craft(A, B).
- If you repeat any action without progress, move to a different scene or approach.

Before choosing an action, silently verify:
- Is the target in the current scene's item list?
- Is the thing I want to use already in my bag?
- For apply(X, Y): Is Y listed under items in the current scene?
- For craft(A, B): Have I already done this craft successfully?
- Have I tried this exact action before and failed?

Please respond in two lines.
Thought: ...
Action: ...
```

Figure 6: Prompt 1. Shared System Prompt (Base / EscapeAgent / Self-Refine).

```
<RECENT_HISTORY>

YOUR OBJECTIVE: <GAME_OBJECTIVE>

Now you need to act on [Step <STEP>]
Your current position is: <POSITION>.

<SCENE_DESCRIPTION>
<POSSIBLE_ACTIONS>
<TOOLS_IN_BAG>

NOTE:
- When you use move, you MUST use exactly one listed scene name.
- Do NOT include extra description such as ": It leads to ...".
- Valid scene names: <VALID_SCENE_NAMES>
- Example: move(<VALID_SCENE_NAME>)

IMPORTANT EXPLORATION RULES:
- If an action fails and nothing changes, do NOT repeat it.
- If the environment gave a MEANINGFUL response, stay with that target and try a DIFFERENT tool or approach.
- If the environment gave NO response ("Nothing happens"), abandon that specific action.
- Avoid repeating the same tool-target interaction.
- Explore different strategies: different tools, crafting, different objects, or another scene.

Your goal is creative exploration, not repeating identical attempts.

<DIVERSITY_CONSTRAINT_IF_ANY>

Your Response:
```

Figure 7: Prompt 2. Base User Prompt Template (Base).

```
DIVERSITY CONSTRAINT (VERY IMPORTANT):
- You are running multiple trials. You MUST discover a DIFFERENT solution each trial.
- You MUST NOT finish the game using any of the following forbidden finish actions:
  * <FORBIDDEN_FINISH_ACTION>
- If you think one of them would solve it, you must intentionally pursue an alternative approach.
```

Figure 8: Prompt 3. Diversity Constraint (All methods).

```
You are currently exploring the scene freely. You should try explore new scenes, interact with the items
      through click, input or apply actions, and try crafting new tools:
- If there's still <interactable items> you haven't tried any action to interact with, you should try 'click'
      them first.
- Otherwise, explore other new <interactable scene> you haven't been to, or going back to parent scene.
- Follow the rules in Possible Actions and system prompt to give a valid action and thought.
Do not repeat actions in history and previous steps. Your Response:
```

Figure 9: Prompt 4. Free Exploration (EscapeAgent).

```
You are a helpful agent to reflect on your action and environment response, and then maintain a task list with
      solving suggestions.
The role of this task list is that there are some tasks you currently cannot solve with the tools at hand, but
      you think you may need to solve them later, so write them down with some suggestions and hints for your
      future reference.

After analyzing your current action and the response from the environment, you should give an action to
      maintain the task list: <function_list>
<param_explain>
For instance, valid task list maintaining action may be: <use_example>.

Reflection functions:
update(updated_feedback)
The parameter should retain the original feedback and add one new hindsight from the current action.

new(task_name, feedback)
The first parameter is a brief task name. The second parameter is what you have to do to solve this task.

delete(index)
If you choose delete, the first parameter is the index of the completed or useless task.

none()
If you choose none, do not give any parameter.
```

Figure 10: Prompt 5. Reflection System (EscapeAgent).

```
Your current position:
<POSITION>

<SCENE_DESCRIPTION>

<POSSIBLE_ACTIONS>

Your action: <ACTION>

Response from the environment:
<ENVIRONMENT_RESPONSE>

Now please make an action call to maintain the task list in one line. Follow the system instruction to extract
      hint and fill in the parameter for the function call.

Your Response:
```

Figure 11: Prompt 6. Reflection User (EscapeAgent).

```
You have to use your creativity to figure out the use of the tool you have just collected.

There are generally two ways about how to use the tool:
1. Combine this tool with another one in your bag to craft a new tool: craft(<collected tool>, <applicable tool
      >).
2. Apply this tool to a target item in a task: apply(<collected tool>, Target Item in a task).

Hints:
1. Pay attention to the task and tool descriptions. Find the connection between them.
2. In Thought, explicitly consider bag items for crafting and task-list targets for applying.
3. In Actions, give zero to multiple craft/apply calls. For apply, give the task index and justify why the tool
       may solve the task.

User template:
You have just collected a new tool:
<collected tool>: <TOOL_NAME>
Description: <TOOL_DESCRIPTION>

Other tools in your bag:
<TOOLS_IN_BAG>

Tasks waiting to be solved:
<TASK_LIST>

Please follow the system prompt to output your Thought and Actions. Analyze thoroughly and be bold to propose
      plausible craft and apply actions.
```

Figure 12: Prompt 7. Forethought for New Tool (EscapeAgent).

```
You have to use your creativity to figure out if you could use any tools you have now to solve a new task you
      have just discovered.

There are generally three ways to solve a task:
1. Click the target item.
2. Apply a bag tool to the target item.
3. Input a string to the target item.

Hints:
1. Pay attention to what the task needs. Always first try simple click if not done.
2. Examine tool descriptions and memory-pad hints to connect them to the task.
3. In Thought, consider click, apply, and input possibilities.
4. In Actions, give zero to multiple click/apply/input calls and justify each.

User template:
The current task:
[Task] Name: <TASK_NAME>, Target Item: <TARGET_ITEM>
<TASK_DESCRIPTION>

Tools in your bag:
<TOOLS_IN_BAG>

Hints from the memory pad:
<MEMORY_PAD>

Please follow the system prompt to output your Thought and Actions. Analyze thoroughly and be bold to propose
      plausible click, apply, and input actions.
```

Figure 13: Prompt 8. Forethought for New Task (EscapeAgent).

```
SELF-REFLECT FEASIBILITY CHECK

Before choosing the final action, silently perform a step-wise feasibility check:
- Is the action physically feasible?
- Does it use only objects explicitly listed in Possible Actions or Tools in Bag?
- Is the action grammar valid?
- For apply(tool, item), is the tool already in the bag and is the target an item in the current scene?
- For craft(tool A, tool B), are both tools already in the bag?
- For move(scene), is the scene name one of the valid listed scene names?
- Does the action plausibly move toward the objective?
- Does it avoid repeating a clearly failed attempt from the history?

If your first candidate is invalid or weakly grounded, revise it before writing the final answer.
Do not show a separate verification section. Only output the final checked action.

Thought: ...
Action: ...
```

Figure 14: Prompt 9. Self-Refine Suffix (Self-Refine).

```
You are a one-shot feedback-guided creative reasoning module for a text environment.

The normal action policy has been temporarily replaced because the agent has accumulated feedback from failed
      interactions with the same target item.
Your job is to use a Divergent-Convergent process to choose the NEXT action.

Game action grammar:
- apply(<tool in your bag>, <interactable item in the current scene>)
- input(string, <interactable item in the current scene>)
- craft(<base tool in your bag>, <ingredient tool in your bag>)

Operational rules:
- Use only names explicitly listed in Possible Actions or Tools in Bag.
- apply() can only use a bag tool as the first argument and a current-scene item as the second argument.
- craft() can only combine two bag tools. Order matters.
- input() can only target a current-scene item.
- The final action must be apply(), input(), or craft(). Do not choose click() or move().

Think in two phases:
1. Divergent phase: generate diverse candidate mechanisms and action sketches. Do not choose the final action
      yet.
2. Convergent phase: apply the objective, valid actions, and accumulated failure evidence to choose exactly one
       valid next action.

Use only creative interaction mechanisms:
1. Try a new tool-based mechanism on the same target with apply().
2. Try an evidence-based input mechanism on the same target with input().
3. Craft tools to create a new capability that could address the target or objective.

Thought: ...
Action: ...
```

Figure 15: Prompt 10. ReDNA System Prompt (ReDNA).

```
FEEDBACK-GUIDED R/DN MODULE

Current step: <STEP>
Current position: <POSITION>
Feedback target item: <TARGET_ITEM>
Failure count on this target: <FAILURE_COUNT>
Non-click failure count on this target: <NON_CLICK_FAILURE_COUNT>

DIVERGENT PHASE INPUTS
Target item:
<TARGET_ITEM>

Current scene context:
<SCENE_DESCRIPTION>

Tools in Bag:
<TOOLS_IN_BAG>

DIVERGENT PHASE
Generate 2-4 substantially different candidate mechanisms using apply(), input(), or craft().
Consider:
- apply(tool, target item): apply a bag tool to the target item in a non-obvious way.
- input(string, target item): try an evidence-based code, word, label, or sequence.
- craft(tool A, tool B): combine two bag tools before returning to the target.
Do not propose click() or move().

CONVERGENT PHASE INPUTS
Objective:
<GAME_OBJECTIVE>

Accumulated failed attempts:
<FAILED_ATTEMPTS_WITH_RESPONSES>

Possible Actions:
<POSSIBLE_ACTIONS>

<DIVERSITY_CONSTRAINT_IF_ANY>

CONVERGENT PHASE
Choose exactly one valid next action.
Rules:
- Do not repeat any failed action exactly.
- Reject candidates contradicted by failure responses.
- Use accumulated feedback to change the failed mechanism.
- Choose a valid, non-repeated apply(), input(), or craft() action.
- If choosing input(), the string should be evidence-based.
- If choosing craft(), explain why the two tools combine into a useful capability.
- The final Action must start with apply(, input(, or craft(.

Before writing the final response, explain what the latest failure implies, compare 2-4 concrete candidates,
      and choose the final action because it addresses what the failed action lacked.

Thought: ...
Action: ...
```

Figure 16: Prompt 11. ReDNA User Prompt Template (ReDNA).