AGENTMERCURY: YOUR AGENT CAN SYNTHESIZE VERIFIABLE ENVIRONMENTS FOR BUSINESS SCENARIOS AT SCALE

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents learn to act through interaction with environments, yet the environments used for training are often manually constructed or synthesized around predefined tasks and benchmarks. This task-centric paradigm makes it difficult to scale environments that reflect realistic workflows where diverse tasks can naturally emerge from the underlying world. We introduce AGENTMERCURY, a scalable framework for synthesizing executable environments from high-level business scenarios. Rather than constructing an environment for a specific task, AGENTMERCURY first instantiates a persistent world with entities, services, tools, state, and executable cross-service invariants, from which diverse tasks can emerge. We construct 4,783 executable environments spanning 14 industries and 50 countries, 4,326 of which are trainable and serve as substrates for reinforcement learning. Although they are generated without targeting the evaluation benchmarks, policies trained on them improve on both enterprise workflows and out-of-domain benchmarks spanning reasoning, coding, scientific computing, and tool use: Qwen3.5-4B improves from 12.3 to 15.7 on ENTERPRISEOPS-GYM and from 45.9 to 56.0 on AIME26. The construction process itself can also be learned: fine-tuning Qwen3.5-35B-A3B on construction traces increases executable-world authoring success from 3.3% to 83.3% on held-out business scenarios. We release the synthesized environments and tasks, construction code, and trained policy models.

1 INTRODUCTION

Modern agentic systems learn to interact with environments through complementary capabilities for acting and modeling the world (Nakano et al., 2021; Yao et al., 2023). A *policy model* learns how to select actions given the agent's current state, while a *world model* predicts how the environment evolves in response to those actions (Sutton & Barto, 1998; Ha & Schmidhuber, 2018; Zuo et al., 2026). Yet both typically assume that the world already exists: the entities, services, tools, state, and transition structure of the environment are specified before the agent begins learning (Song et al., 2026; Xu et al., 2026b; Wang et al., 2026; Dong et al., 2026; Zuo et al., 2026). We argue that this leaves a third role largely outside the learning loop: *constructing the world itself* (Figure 2). We distinguish three roles: (1) a PLANET role that determines what world exists, (2) a policy π that determines what the agent does within that world, and (3) a world model W that models how the world responds. We therefore ask: *how can we systematically construct executable worlds in which agents learn and operate?*

Environments are typically designed around the tasks that agents are expected to solve. Manually constructed environments specify entities, tools, states, and transition logic for a particular domain or task (Shridhar et al., 2021; Zhou et al., 2024; Xie et al., 2024; Drouin et al., 2024), while synthetic environments are often generated from task descriptions, user instructions, or evaluation specifications (Liu et al., 2024a; Dong et al., 2026; Zuo et al., 2026; Kimi Team et al., 2026). This task-centric paradigm has driven the rapid development of increasingly capable benchmarks (Jimenez et al., 2024; Li et al., 2026; Mavali et al., 2026). However, it couples the construction of a world to the tasks used to define or evaluate it: the resulting environment is often optimized to make a particular task executable or measurable, rather than to represent a broader scenario from which diverse

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

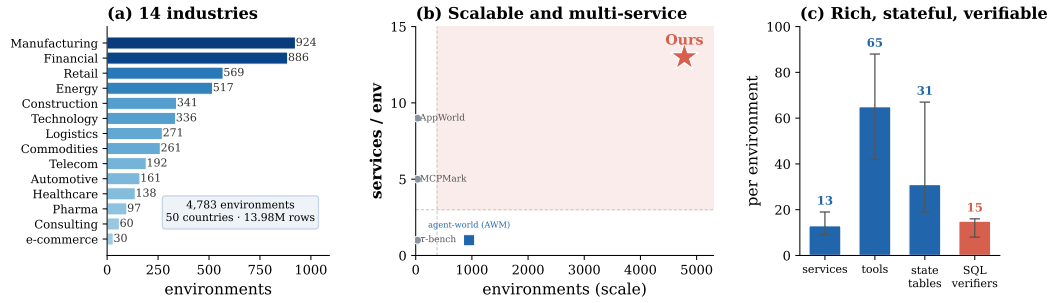

Figure 1: **AGENTMERCURY synthesizes diverse, verifiable business environments at scale.** **(a)** 4,783 environments spanning 14 industries and 50 countries; every company name is replaced with a substitute name to prevent hallucination. **(b)** Environment scale and multi-service depth compared with existing agent environments (Trivedi et al., 2024; Wu et al., 2026; Shi et al., 2026; Dong et al., 2026); AGENTMERCURY occupies the regime that is both large-scale and multi-service. **(c)** Each environment contains multiple services, tools, state tables, and deterministic SQL-based verifiers that check cross-service constraints.

tasks can naturally emerge (Cobbe et al., 2020). Increasing the number of tasks therefore does not necessarily increase the diversity of the underlying worlds (Wang et al., 2026).

Real-world agent applications (He et al., 2024; Yao et al., 2022) instead require environments that capture ongoing workflows, such as SWE (Deng et al., 2025) and τ-bench-style domains (Barres et al., 2025; Shi et al., 2026); in such settings, a task emerges from an evolving workflow involving user intents, persistent state, software services, and interactions across multiple systems (Qin et al., 2024; Li et al., 2023; Bandi et al., 2026; Wu et al., 2026; Zheng et al., 2026). A sufficiently rich world can therefore support many distinct tasks and interaction trajectories without requiring each task to be specified during world construction (Cheng et al., 2026; Liu et al., 2026). This motivates a shift from *task-centric environment construction* toward *scenario-grounded world generation* (Zeng et al., 2026; Kimi Team et al., 2026), where a high-level scenario is the source from which an executable world and its possible tasks can emerge. Business workflows provide a natural setting because they combine structured operational processes with diverse interactions among users, software systems, tools, and evolving persistent state.

We introduce AGENTMERCURY, a scalable framework for authoring executable environments from high-level business scenarios. Rather than starting from individual tasks, PLANET first instantiates the world (its entities, services, tools, persistent state, transition dynamics, and world-level invariants), from which diverse tasks and trajectories then arise. The invariants are rendered as executable verification conditions rather than hard-coded as transition rules, so the same world can support different tasks while providing deterministic signals for evaluating their outcomes. Separating world construction from task instantiation lets environments be generated independently of any particular benchmark task and reused across training and evaluation. As shown in Figure 1, AGENTMERCURY synthesizes 4,783 executable environments spanning 14 industries and 50 countries, 4,326 of which are trainable, competitive in scale with previous works (Trivedi et al., 2024; Wu et al., 2026; Shi et al., 2026; Dong et al., 2026) while providing substantially richer multi-service and stateful interactions.

We first investigate whether these environments can serve as effective training substrates for policy learning. We train policies with reinforcement learning directly in AGENTMERCURY worlds, where they must interact with persistent state, use available tools, and satisfy constraints induced by the underlying business scenarios. The resulting policies improve not only on these training tasks but also on established benchmarks that were not used to construct the training environments, spanning enterprise workflows as well as out-of-domain reasoning, coding, scientific computing, knowledge, and tool use. These results suggest that scaling the diversity of scenario-grounded worlds can provide learning signals that extend beyond the specific tasks used during training.

We next ask whether the construction process itself can become a learnable capability. AGENTMERCURY exposes the intermediate construction traces used to transform business scenarios into executable worlds, including changes to entities, services, persistent state, transition logic, and ver-

ification conditions. Fine-tuning a policy model on these traces substantially improves its success rate in authoring executable worlds from unseen business scenarios, showing that environment construction is not only an engineering process performed outside the learning loop but also a learnable capability. This points toward a scalable loop between scenario-grounded world generation and agent learning.

In summary, our contributions are as follows: (1) We introduce AGENTMERCURY, a scalable framework for scenario-grounded synthesis of executable environments that makes the construction of the agent's world an explicit PLANET role. (2) We construct 4,783 executable business environments spanning 14 industries and 50 countries and release them with their business scenarios and task instructions, construction code, and trained policy models. (3) We show that policies trained in these environments improve on in-domain enterprise workflows and on out-of-domain benchmarks, although the environments are constructed independently of these benchmarks. (4) We show that the construction traces generated by AGENTMERCURY provide effective supervision for environment authoring, suggesting that world construction can become a learnable capability.

2 PRELIMINARIES

A policy π specifies how an agent acts within an environment: at timestep t, it maps the interaction history h_t to a distribution over actions, but does not determine what entities, services, or transition mechanisms constitute the world (Sutton & Barto, 1998). An environment is an executable representation of a business scenario in which an agent directly interacts with software services (Zhou et al., 2024; Drouin et al., 2024). The environment maintains a state s_t of the underlying software, including persistent data and service-level state (Yao et al., 2025; Xie et al., 2024); the agent does not observe s_t directly. The agent acts through tool calls to available services and receives their outputs as observations ω_{t+1} from an observation function O:

$$a_t \sim \pi(\cdot \mid h_t), \qquad s_{t+1} = T(s_t, a_t), \qquad \omega_{t+1} \sim O(\cdot \mid s_{t+1}, a_t). \tag{1}$$

Here, T is the executable transition function implemented by the environment, whereas a world model $W(s_{t+1} \mid s_t, a_t)$ learns to approximate these dynamics (Ha & Schmidhuber, 2018; Hafner et al., 2019; 2023; Schrittwieser et al., 2020): it models *how an existing world evolves*, rather than determining *what world exists*. Environment execution presupposes a specification: the entities, services, initial state, transition logic, and constraints must be determined before an agent can interact with the resulting world. This separation between constructing a world and acting within it motivates the additional role introduced in Section 3.

3 METHOD

AGENTMERCURY treats environment construction as an explicit role within the agent–environment system. Given a high-level business scenario σ, the system first constructs an executable world and then instantiates tasks from it; agents interact with the world through executable tools, and their trajectories are evaluated against deterministic task- and world-level constraints:

$$\sigma \xrightarrow{\text{PLANET}} w \xrightarrow{\text{TASK}} (u, \rho) \xrightarrow{\pi} \tau \xrightarrow{\text{GRADE}} r, \tag{2}$$

where w denotes an executable world, u is a task instruction, ρ is its task-specific grading specification, τ is an interaction trajectory, and r is the resulting reward. This decomposition separates world construction from task specification: the world is generated once from a scenario, while multiple tasks and trajectories can subsequently emerge from the same underlying world (Figure 2(A)).

3.1 PLANET: SCENARIO-TO-WORLD CONSTRUCTION

Existing agent environments are typically provided as fixed artifacts, constructed manually or through task-conditioned synthesis (Song et al., 2026; Xu et al., 2026b;a). In contrast, we introduce PLANET as an explicit environment-authoring role that constructs an executable world $w \sim \text{PLANET}(\cdot \mid \sigma)$ from a high-level business scenario σ; in our implementation, environment synthesis uses DeepSeek-V4-Flash (DeepSeek-AI, 2026). We represent an executable world as

$$w = \langle \mathcal{S}, \mathcal{A}, \Omega, T, O, s_0, \mathcal{R} \rangle, \tag{3}$$

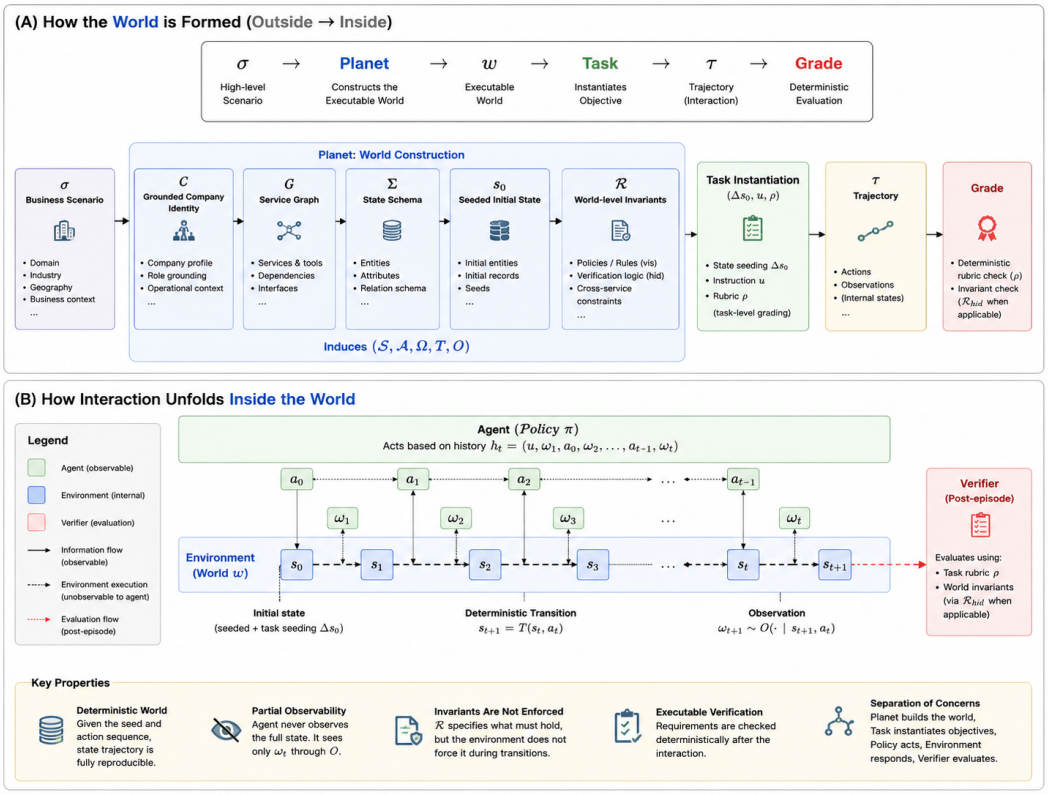

Figure 2: **From scenario-grounded world construction to agent interaction. (A)** PLANET transforms a business scenario σ into an executable world w with a company identity, service graph, state schema, seeded initial state s_0, and world-level invariants $\mathcal{R}$; a task is then instantiated through task-specific state seeding and a rubric. **(B)** A policy acts from the interaction history; its actions deterministically transition the environment from s_t to s_{t+1}, and observations ω_{t+1} are exposed through the observation interface. The trajectory is evaluated post-episode with the task rubric and, where applicable, the invariants.

where $\mathcal{S}$, $\mathcal{A}$, and Ω denote the state, action, and observation spaces, T is the executable transition function, O is the observation function, and s_0 is the seeded initial state. The set $\mathcal{R}$ denotes the world-level invariants: properties that the world requires to hold but does not itself enforce during state transitions.

Structured World Synthesis. PLANET decomposes scenario grounding into structured artifacts: the grounded company identity C, the service graph G, and the state schema Σ. The world-generation process factorizes as

$$\begin{aligned}\textsc{Planet}(w \mid \sigma) = p(C \mid \sigma)\, p(G \mid C)\, p(\Sigma \mid G, C) \\ \times\, p(s_0 \mid \Sigma)\, p(\mathcal{R} \mid G, \Sigma, s_0),\end{aligned} \tag{4}$$

where $(\mathcal{S}, \mathcal{A}, \Omega, T)$ are determined by the resulting service graph and state schema. This decomposition separates the construction of the world structure from the construction of its invariant set.

The invariant set $\mathcal{R}$ is not part of the transition mechanism: T applies agent actions to the underlying software state, whereas $\mathcal{R}$ defines conditions that can subsequently be checked against that state. This distinction allows the environment to represent workflows in which the agent is responsible for satisfying cross-service requirements rather than having those requirements automatically enforced by the simulator. For example, a cross-service invariant may require a downstream record to exist after an upstream business event, while the environment does not automatically create that record on the agent's behalf. The same invariants appear in two views: a visible view $\mathcal{R}_{\text{vis}}$, surfaced through in-world documents or policies that the agent must discover through interaction rather than being

given as an oracle, and a hidden view $\mathcal{R}_{\text{hid}}$ that contains executable verification conditions used for deterministic evaluation.

3.2 Task Instantiation from a World

Once an executable world has been constructed, tasks are generated from the resulting world rather than used as the specification from which the world itself is built. A task generator receives the initial world state and its invariants and produces a user objective together with a task-specific grading specification:

$$(\Delta s_0, u, \rho) \sim \textsc{Task}(\cdot \mid s_0, \mathcal{R}), \tag{5}$$

where Δs_0 denotes task-specific state seeding applied to the world before interaction begins and ρ is the task rubric. Multiple tasks may therefore share the same world structure while starting from different task-specific states. The rubric ρ specifies the task-level assertions required for successful completion, while $\mathcal{R}$ continues to describe invariants of the underlying world; this separation prevents task-specific objectives from being conflated with the rules that define the world itself. Appendix A describes the resulting task corpus.

3.3 Interaction, Deterministic Grading, and Policy Learning

The agent then interacts with the executable world following Eq. (1), and each observation is appended to its interaction history, $h_{t+1} = (h_t, a_t, \omega_{t+1})$ (Figure 2(B)). Because the transition function is executed by the environment rather than predicted by a learned model, the resulting trajectories are reproducible under a fixed initial seed and action sequence, and can be replayed and deterministically re-scored after interaction (Appendix B). We do not train a separate world model in this work: AgentMercury uses the executable environment itself as the source of transition dynamics.

Given a completed trajectory τ of horizon H, the grader evaluates the final state and task-specific assertions against the corresponding rubric:

$$r(\tau) = \textsc{Grade}(s_H, \tau; \rho, \mathcal{R}_{\text{hid}}) \in [0, 1]. \tag{6}$$

Whenever possible, ρ and $\mathcal{R}_{\text{hid}}$ are evaluated through deterministic checks over the underlying database state rather than relying solely on model-based judgment; the reward used for RL additionally includes an LLM judge and a behavior penalty (Appendix C).

Policy Learning on Synthesized Worlds. Unlike task-centric environment construction, the environments used during training are generated from business scenarios independently of the target evaluation tasks. A policy π_θ therefore learns to operate over a distribution of executable worlds containing diverse services, tools, states, workflows, and cross-service constraints, using rewards built on $r(\tau)$ (Eq. (6)). This training loop couples policy learning to executable worlds without requiring the environment generator to enumerate the tasks that the policy will encounter.

4 Experiments

We evaluate AgentMercury with two questions. **Can synthesized worlds support effective and stable policy optimization on business-oriented agent tasks?** This evaluates the world as a training substrate. **Can agents learn to construct executable worlds from high-level business scenarios?** This evaluates whether the construction process exposed by AgentMercury can itself become a learnable agent capability.

4.1 Experimental Setup

We train Qwen3.5-4B and Qwen3.5-35B-A3B (Qwen Team, 2026) with group relative policy optimization (GRPO) (Shao et al., 2024), following the coefficient settings of Dr. GRPO (Liu et al., 2025b). To show that our environments do not depend on a specific RL algorithm, we also apply single-rollout asynchronous optimization (SAO) (Hou et al., 2026). The training corpus contains 43,300 task instances generated from the 4,326 trainable environments among the 4,783 synthesized ones, i.e., those that pass task generation and verification (Appendix A). We evaluate the resulting policies on EnterpriseOps-Gym (Malay et al., 2026), a business-oriented agent benchmark

Table 1: Results on ENTERPRISEOPS-GYM (Malay et al., 2026) for Qwen3.5-4B and Qwen3.5-35B-A3B before and after training on environments and tasks synthesized by AGENTMERCURY, with GRPO (Shao et al., 2024) or SAO (Hou et al., 2026) as the RL algorithm. † denotes results derived from ENTERPRISEOPS-GYM.

Model	Teams	CSM	Email	ITSM	Calendar	HR	Drive	Hybrid	Avg.
GPT-5 †	26.3	36.4	49.0	18.9	41.3	17.9	34.0	23.5	30.9
Gemini-2.5-Pro †	39.3	11.6	31.1	13.9	12.5	4.9	27.0	19.6	19.9
Kimi-K2-Thinking †	30.0	7.1	51.0	12.2	15.4	8.2	39.6	15.7	22.4
Qwen3-4B (Think) †	24.0	3.8	38.4	5.6	5.8	7.1	21.9	15.8	15.3
Qwen3-30B (Think) †	22.0	5.4	51.9	6.7	18.3	7.6	25.7	15.7	19.1
Qwen3-235B (Think) †	28.0	4.7	38.1	9.3	15.7	7.8	23.8	17.7	18.1
Qwen3.5-4B	20.8±2.5	9.2±2.3	23.9±3.0	6.8±1.0	10.4±2.5	9.5±2.8	6.2±1.6	11.7±1.7	12.3±0.2
Qwen3.5-4B + GRPO + OURS	23.0±1.6	5.6±1.5	33.3±0.9	7.4±1.1	13.1±1.6	10.8±4.3	15.6±3.1	17.0±1.1	15.7±0.6
Qwen3.5-35B-A3B	33.9±3.8	7.6±2.5	53.2±2.3	14.6±1.0	18.6±0.9	11.4±1.5	35.9±5.4	23.5±1.7	24.8±0.6
Qwen3.5-35B-A3B + GRPO + OURS	39.8±4.1	11.1±0.9	53.8±7.5	15.6±1.6	21.9±1.9	17.7±0.9	41.2±3.9	23.7±1.1	28.1±1.4
Qwen3.5-35B-A3B + SAO + OURS	39.1±2.3	12.5±2.8	54.8±2.1	16.5±0.5	22.6±0.5	17.7±1.4	39.1±4.4	24.1±2.4	28.3±1.5

Table 2: Out-of-domain evaluation across mathematical reasoning, coding, scientific computing, tool use, and agentic benchmarks, before and after RL training on tasks synthesized by AGENTMERCURY. We report mean ± standard deviation over three independent evaluation runs.

Model	AIME26	HMMT	LCB	SciCode	τ^3-Airline	τ^3-Retail	τ^3-Telecom	BFCL	GPQA-Diamond
Qwen3.5-4B	45.9±1.7	28.5±1.2	36.6±2.8	22.6±0.1	48.8±2.1	70.4±1.0	92.5±1.0	30.3±1.8	76.5±0.7
Qwen3.5-4B + GRPO + OURS	56.0±1.8	35.4±1.6	44.0±1.2	25.7±0.7	58.7±0.8	73.6±1.3	91.9±1.3	31.7±1.5	77.5±0.6
Qwen3.5-35B-A3B	91.0±0.3	77.0±0.9	74.3±2.2	29.7±1.5	39.1±10.6	52.7±22.1	49.1±49.8	31.1±1.3	82.8 ±3.0
Qwen3.5-35B-A3B + GRPO + OURS	91.9±2.1	80.0±0.7	79.0±0.5	29.9±1.7	52.5±5.2	57.1±8.3	68.9±20.5	42.5±0.3	85.0±0.5
Qwen3.5-35B-A3B + SAO + OURS	92.2±1.9	83.3±3.1	78.6±1.0	28.3±1.9	50.9±8.1	55.7±10.2	65.5±23.8	42.1±0.1	84.0±1.0

covering multi-service workflows and tool-based interactions, and on out-of-domain benchmarks: AIME26 (Mathematical Association of America, 2026), HMMT (Dekoninck et al., 2026), LiveCodeBench v5–v6 (Jain et al., 2025), SciCode (Tian et al., 2024), τ^3 (Shi et al., 2026), BFCL (Patil et al., 2025), and GPQA-Diamond (Rein et al., 2023). For each benchmark, we report the mean and standard deviation over three independent evaluation runs. Our training environments are constructed independently of all benchmark tasks, so these evaluations test whether scenario-grounded environments provide transferable learning signals rather than merely reproducing a target benchmark. Hyperparameters and benchmark details are in Appendices C and D.

4.2 POLICY OPTIMIZATION OF BUSINESS-ORIENTED TASKS

In Table 1, GRPO training on AGENTMERCURY raises the average score of Qwen3.5-4B from 12.3 to 15.7 (+3.4 points, +27.6%). The largest gains are in Drive (6.2 → 15.6) and Email (23.9 → 33.3), both +9.4, followed by Hybrid (11.7 → 17.0). Performance on CSM decreases from 9.2 to 5.6, but the improvement across seven of the eight domains indicates that training on AGENTMERCURY-synthesized environments transfers to unseen enterprise workflows. The improvement also extends beyond GRPO to SAO (Hou et al., 2026) at the larger model scale. For Qwen3.5-35B-A3B, GRPO increases the average score from 24.8 to 28.1 (+3.3 points, +13.3%) and SAO to 28.3 (+3.5 points, +14.1%). Both improve all eight domains, with the largest gains on HR (+6.3 for both) and Teams (+5.9 and +5.2), although some gains are small relative to run-to-run variation (e.g., Email 53.2 → 53.8 and Hybrid 23.5 → 23.7 under GRPO; Appendix E). These results suggest that the learning signal provided by AGENTMERCURY-synthesized environments is not tied to a single policy optimization algorithm, particularly as the policy model scales (Section 6).

4.3 POLICY OPTIMIZATION OF BENCHMARK-ORIENTED TASKS

As shown in Table 2, training on AGENTMERCURY improves Qwen3.5-4B across substantially different capability domains: from 45.9 to 56.0 on AIME26, 28.5 to 35.4 on HMMT, and 36.6 to 44.0 on LiveCodeBench. The same trend extends to SciCode (22.6 → 25.7), BFCL (30.3 → 31.7), GPQA-Diamond (76.5 → 77.5), and two of the three τ^3 domains (Airline, 48.8 → 58.7; Retail, 70.4 → 73.6). Telecom remains largely unchanged (92.5 → 91.9), suggesting that the transfer is not simply a uniform reward-driven shift across all benchmarks. The transfer from business-oriented tool-use trajectories to mathematical reasoning, coding, scientific computing, and independent tool-

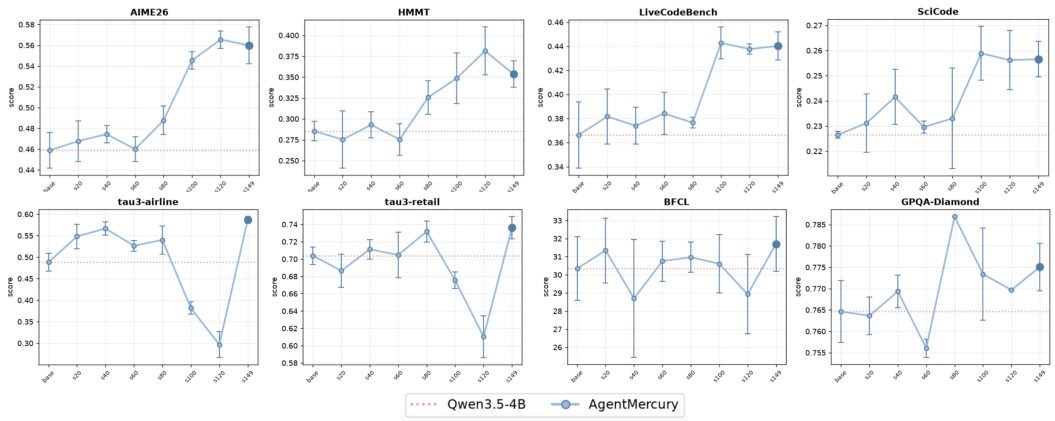

Figure 3: Out-of-domain benchmark performance across training checkpoints for Qwen3.5-4B + GRPO + OURS. The dashed line denotes the base model; error bars show the standard deviation over three independent evaluation runs.

use benchmarks suggests that the synthesized tasks provide a broader policy-learning signal rather than merely inducing benchmark-specific behavior.

At 35B scale, the gains are smaller on already-saturated reasoning benchmarks but remain substantial for tool-oriented capabilities. SAO improves AIME26 from 91.0 to 92.2, HMMT from 77.0 to 83.3, LiveCodeBench from 74.3 to 78.6, and BFCL from 31.1 to 42.1, while SciCode does not improve (29.7 to 29.9 with GRPO and 28.3 with SAO). On τ^3, the base Qwen3.5-35B-A3B model exhibits substantial run-to-run variance because individual runs occasionally collapse, especially on Telecom (49.1 ± 49.8) and Retail (52.7 ± 22.1); after training with SAO, these become 65.5 ± 23.8 and 55.7 ± 10.2, while Airline improves from 39.1 ± 10.6 to 50.9 ± 8.1. The benefit of synthesized environments is not limited to raising the expected reward: the learned policy also becomes considerably more consistent across independently sampled interactions, which matters for agentic tasks where success depends on a sequence of tool calls and state transitions.

Figure 3 tracks Qwen3.5-4B + GRPO + OURS across training checkpoints on the same out-of-domain benchmarks. The improvement is progressive rather than concentrated at a single checkpoint: mathematical reasoning, competitive programming, and scientific computing capabilities generally improve as policy optimization proceeds. This trend provides evidence that these gains are a consequence of continued policy learning from the synthesized environments rather than evaluation noise at a single checkpoint. Training dynamics (Appendix E.2) indicate that the reward improvement is not driven by response degeneration: at both model scales, reward increases while the truncated-response ratio decreases and the degenerate-response ratio remains near zero, apart from a small late spike in the 35B SAO run.

4.4 CAN AGENTS LEARN TO AUTHOR EXECUTABLE WORLDS?

We next ask whether a model given a high-level business scenario can construct an executable world that satisfies the required structural constraints, including services, state schemas, tools, and cross-service invariants. We construct a held-out set of 30 synthetic business briefs sampled from the country–industry distribution of the environment library; each brief specifies a business scenario without exposing the underlying company environment, and none of these briefs or their worlds appear in the fine-tuning data used below. We evaluate four off-the-shelf API models in two settings: zero-shot (brief only) and recipe-conditioned (brief plus an invariant digest and a trimmed example of a previously constructed environment). We grade authoring by an executable oracle rather than by surface-level similarity to a reference: a world counts as successful only when it passes all 12 structural validators adapted from the AGENTMERCURY construction oracle, so it must be executable and structurally consistent.

Environment Authoring Performance. Table 3 shows that executable-world authoring is feasible for sufficiently capable models even without explicit construction examples. The four API models

Table 3: Oracle-pass rates (all 12 structural validators passed) for authoring executable worlds from 30 held-out business briefs. Recipe adds an invariant digest and a trimmed construction exemplar to the brief.

Model	Zero-shot	Recipe
Claude Opus 4.8	83.3%	80.0%
GPT-5.4	66.7%	66.7%
DeepSeek-V4-Pro	83.3%	86.7%
GLM-5.2	80.0%	76.7%
Mean (API models)	78.3%	77.5%
Qwen3.5-35B-A3B	3.3%	20.0%
Qwen3.5-35B-A3B + AGENTMERCURY	**83.3%**	10.0%

achieve zero-shot oracle-pass rates between 66.7% and 83.3%, with a mean of 78.3%. Providing an explicit construction recipe does not consistently improve performance: the mean pass rate decreases slightly to 77.5%, and the paired comparison on the same 30 briefs shows no systematic advantage. The remaining failures are concentrated in constraints that require reasoning across multiple services. A common failure mode is the collapse of a cross-service constraint, where the trigger and target of an invariant are placed within the same service; this affects 10 of the 30 briefs for GPT-5.4. Such errors cannot be detected reliably from the specification alone, since the world may still appear syntactically valid; we therefore view the executable oracle as an essential part of authoring.

Learning the Authoring Process. The results change substantially when the construction traces themselves are used as supervision. We fine-tune Qwen3.5-35B-A3B on 29,823 samples derived from AGENTMERCURY construction traces, covering brief-to-world generation, intermediate-stage completion, validator-guided corruption repair, and intent-to-diff prediction. Before fine-tuning, the base model passes the full oracle on only 3.3% of the held-out briefs and frequently truncates or breaks the required output format. After training, the oracle-pass rate increases to 83.3% (Fisher exact test, $p = 1.2 \times 10^{-10}$), matching the strongest off-the-shelf API models; the trained model passes 11.5 of the 12 validators on average, and truncation and formatting failures are nearly eliminated. These results provide evidence that environment authoring is not merely an engineering procedure external to the agent: AGENTMERCURY provides both worlds that can serve as training substrates and construction traces that can teach agents how to create such worlds.

Recipe conditioning behaves differently before and after training. The recipe improves the base model from 3.3% to 20.0% but reduces the fine-tuned model from 83.3% to 10.0%, with 27 of the 30 recipe-conditioned generations failing the cross-service validation check. The recipe can partially compensate for the missing construction capability of the base model, whereas after fine-tuning, the construction procedure is internalized in the model parameters and the additional recipe can interfere with the learned generation policy. Details are in Appendix F.

5 RELATED WORK

Environment Synthesis for Agent Training. Recent systems synthesize executable environments for agentic RL at scale, including programmatically synthesized tool environments (Song et al., 2026; Xu et al., 2026b) and real-world environments (Dong et al., 2026). Others synthesize infinite synthetic worlds (Wang et al., 2026) and terminal- and phone-use environments (Cheng et al., 2026; Liu et al., 2026), on the thesis that environment diversity rather than trajectory count drives generalization (Zhang et al., 2026). Several of these expose a single service per environment (Wang et al., 2026; Cheng et al., 2026; Liu et al., 2026), so a task never has to coordinate across services. AGENTMERCURY instead constructs multi-service business worlds before any task exists, couples their services through cross-service invariants, and treats the construction process itself as learnable.

Agent Benchmarks. Hand-built interactive suites span embodied, web, computer-use, software, and app environments (Shridhar et al., 2021; Deng et al., 2023; Zhou et al., 2024; Xie et al., 2024; Jimenez et al., 2024; Trivedi et al., 2024); the τ-bench line adds policy-driven customer service (Yao

et al., 2025; Barres et al., 2025; Shi et al., 2026), tool-use benchmarks add breadth (Li et al., 2023; Mialon et al., 2024; Patil et al., 2025; Li et al., 2026), and a recent line builds directly on MCP servers (Luo et al., 2025; Wu et al., 2026; Bandi et al., 2026). Each is hand-built and therefore a fixed testbed: enough to evaluate on, not to train on at scale.

Synthetic Tool-Use Data and Construction as Software Engineering. Building on instruction self-generation (Wang et al., 2023), tool-learning corpora connect models to large API collections (Qin et al., 2024; Patil et al., 2024) and generate function-calling data at scale (Liu et al., 2024b; 2025a; Prabhakar et al., 2025). These supply volume, yet the tasks are largely single-call or stateless; we take up stateful, cross-service work. SWE-bench (Jimenez et al., 2024) grades code edits by fail-to-pass tests, and a growing line synthesizes such instances at scale (Jain et al., 2024; Yang et al., 2025); we apply this idea to our own construction pipeline, whose traces supervise world authoring graded by executable structural validators (Section 4.4).

6 Discussion and Conclusion

What should we scale? Effective environment scaling does not necessarily require constructing environments around the target benchmarks. AgentMercury generates environments from high-level business scenarios, without access to the task instances used by our evaluation benchmarks, yet policies trained in these worlds improve both on EnterpriseOps-Gym and on out-of-domain reasoning, coding, scientific computing, and tool-use benchmarks. This suggests that diverse executable worlds can expose policies to reusable patterns of interaction, state tracking, constraint satisfaction, and tool-mediated decision making that transfer across task distributions. We argue that future systems should investigate how to scale the diversity, structure, and realism of the underlying worlds rather than primarily the number of benchmark-specific tasks. Because AgentMercury separates the world, the task, and the policy, a single synthesized world can be reused across multiple objectives and policy-training configurations, and the environment distribution itself can become a target of scaling.

Limitations and Future Work. Our current system does not yet fully realize the closed loop between environment generation and agent learning. Environment synthesis does not employ a learned world model to predict how candidate worlds behave under agent interaction, and the generator cannot yet use the policy's experience to identify which scenarios are most needed for further learning. We also found that SAO was substantially less effective for the 4B model, so we exclude the 4B SAO result from the main comparison rather than treating the unstable optimization as evidence against the environment distribution (Appendix E.3). A natural next step is an agent that monitors its own failures, proposes scenarios targeted at those failures, and feeds the resulting verified worlds back into policy optimization as an adaptive curriculum.

Conclusion. AgentMercury scales persistent, executable worlds grounded in high-level business scenarios rather than isolated task instances around fixed benchmarks. We synthesize 4,783 environments spanning 14 industries and 50 countries, each with persistent state, multi-service interactions, and executable cross-service constraints. Qwen3.5-4B trained with GRPO in these worlds improves from 12.3 to 15.7 on EnterpriseOps-Gym and from 45.9 to 56.0 on AIME26, although the training environments were constructed independently of these evaluation tasks. The construction process itself can also be learned: fine-tuning Qwen3.5-35B-A3B on construction traces increases executable-world authoring success from 3.3% to 83.3% on held-out business scenarios. These results show that scenario-grounded environments can serve both as scalable training substrates for policies and as structured supervision for learning how executable worlds are constructed. This motivates treating Planet as a first-class, learnable component of the agent system: while policies learn how to act within a world and world models learn how the world responds, Planet can determine which worlds should exist and how they should evolve.

AI Use Statement

Generative models are part of the method studied in this paper: AgentMercury synthesizes its executable environments with DeepSeek-V4-Flash (DeepSeek-AI, 2026) (Section 3.1), the authoring study in Section 4.4 evaluates off-the-shelf models as environment authors, and LLM judges are

used in the RL reward and in the τ^3 evaluation (Appendices C and D). Beyond the method itself, we used generative AI tools to implement parts of the code (implementing methods), to create or edit figures, to edit and condense the manuscript, and to format LaTeX and references. We have not used generative AI tools to design the experiments, interpret the results, or clean the datasets. We have reviewed all AI-assisted work, and we take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics Statement

All environments released with this work are synthetic. Company identities are not seeded from real companies, and the entities, employee records, personas, and business data inside each environment are generated by DeepSeek-V4-Flash, so the environments contain no data collected from real individuals or organizations, and no human subjects were involved in constructing or evaluating them. Two risks remain. First, generated worlds inherit the priors of the generator model and may reflect stereotypes about countries or industries, or contain inaccurate business and regulatory details; they are training substrates, not factual references. Second, the tasks train agents to modify state across business systems, and many of them carry cross-system action risk (Appendix A); our policies act only on sandboxed services, and deploying such agents on real enterprise systems would require safeguards such as scoped permissions and human confirmation of consequential actions.

Reproducibility Statement

Appendix A describes the synthesized task corpus and how tasks are sampled during RL. Appendix B gives the full formal description of the agent–environment interaction and grading. Appendix C lists the training configuration of the Qwen3.5-4B GRPO run and describes the SAO setup, and Appendix D specifies the evaluation protocol for every benchmark, including the number of independent runs and samples per problem. Appendix F details the authoring protocol, the oracle-pass criterion, and the statistical tests. We release the synthesized environments and tasks, the construction code, and the trained policy models.

References

Chaithanya Bandi, Razvan-Gabriel Dumitru, Ben Hertzberg, Divyansh Agarwal, Geobio Boo, Tejas Polakam, Sami Hassaan, Jeff Da, HiJae Kim, Vipul Gupta, et al. MCP-Atlas: A large-scale benchmark for tool-use competency with real MCP servers. *arXiv preprint arXiv:2602.00933*, 2026.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982. arXiv:2506.07982.

Zihao Cheng, Hongru Wang, Zeming Liu, Xinyi Wang, Xiangrong Zhu, Yuhang Guo, Wei Lin, Jeff Z. Pan, and Yunhong Wang. Terminal-World: Scaling terminal-agent environments via agent skills, 2026. arXiv:2605.20876.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: MathArena as an evaluation platform for mathematics with LLMs. *arXiv preprint arXiv:2605.00674*, 2026. URL https://arxiv.org/abs/2605.00674.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-World: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, Léo Boisvert, Megh Thakkar, Quentin Cappart, David Vazquez, et al. WorkArena: How capable are web agents at solving common knowledge work tasks? *arXiv preprint arXiv:2403.07718*, 2024.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Zhenyu Hou, Yujiang Li, Jie Tang, and Yuxiao Dong. Single-rollout asynchronous optimization for agentic reinforcement learning. *arXiv preprint arXiv:2607.07508*, 2026.

Naman Jain, Manish Shetty, Tianjun Zhang, King Han, Koushik Sen, and Ion Stoica. R2E: Turning any GitHub repository into a programming agent environment. In *International Conference on Machine Learning*, 2024.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The Tool Decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuxian Wang, Zezhong Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Yong Liu, Yasheng Wang, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. ToolACE: Winning the points of LLM function calling. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025a. arXiv:2409.00920.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024a.

Yuxuan Liu, Xin Lai, Junyi Li, Pengyuan Lyu, et al. PhoneWorld: Scaling phone-use agent environments. *arXiv preprint arXiv:2605.29486*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh Murthy, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. APIGen: Automated pipeline for generating verifiable and diverse function-calling datasets. In *Advances in Neural Information Processing Systems 37 (NeurIPS), Datasets and Benchmarks Track*, 2024b. arXiv:2406.18518.

Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. MCP-Universe: Benchmarking large language models with real-world model context protocol servers, 2025. arXiv:2508.14704.

Shiva Krishna Reddy Malay, Shravan Nayak, Jishnu Sethumadhavan Nair, Sagar Davasam, Aman Tiwari, Sathwik Tejaswi Madhusudhan, Sridhar Krishna Nemala, Srinivas Sunkara, and Sai Rajeswar. EnterpriseOps-Gym: Environments and evaluations for stateful agentic planning and tool use in enterprise settings. *arXiv preprint arXiv:2603.13594*, 2026.

Mathematical Association of America. American Invitational Mathematics Examination, 2026. https://maa.org/maa-invitational-competitions/.

Sina Mavali, David Pape, Jonathan Evertz, Samira Abedini, Devansh Srivastav, Thorsten Eisenhofer, Sahar Abdelnabi, and Lea Schönherr. No more, no less: Task alignment in terminal agents. *arXiv preprint arXiv:2605.12233*, 2026.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2311.12983.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. APIGen-MT: Agentic pipeline for multi-turn data generation via simulated agent-human interplay, 2025. arXiv:2504.03601.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2307.16789.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. tau-Knowledge: Evaluating conversational agents over unstructured knowledge. *arXiv preprint arXiv:2603.04370*, 2026.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. EnvScaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

Minyang Tian, Luyu Gao, Shizhuo D Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. SciCode: A research coding benchmark curated by scientists. *Advances in Neural Information Processing Systems*, 2024.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024. Best Resource Paper; arXiv:2407.18901.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent World Model: Infinity synthetic environments for agentic reinforcement learning. *arXiv preprint arXiv:2602.10090*, 2026.

Zijian Wu, Xiangyan Liu, Lingjun Chen, Fanqing Meng, Lingxiao Du, Yiran Zhao, Fanshi Zhang, Yaoqi Ye, Jiawei Wang, Zirui Wang, et al. MCPMark: A benchmark for stress-testing realistic and comprehensive MCP use. In *International Conference on Learning Representations*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 2024.

Frank Fangzheng Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. TheAgentCompany: benchmarking LLM agents on consequential real world tasks. *Advances in Neural Information Processing Systems*, 2026a.

Minrui Xu, Zilin Wang, Mengyi Deng, Zhiwei Li, Zhicheng Yang, Xiao Zhu, Yinhong Liu, Boyu Zhu, Baiyu Huang, Chao Chen, et al. EnvFactory: Scaling tool-use agents via executable environments synthesis and robust RL. *arXiv preprint arXiv:2605.18703*, 2026b.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, 2025.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. arXiv:2406.12045.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Jiayi Zhang, Fanqi Kong, Guibin Zhang, Maojia Song, Zhaoyang Yu, Jianhao Ruan, Jinyu Xiang, Bang Liu, Chenglin Wu, and Yuyu Luo. Scalable environments drive generalizable agents, 2026. Position paper; arXiv:2605.18181.

Weihuang Zheng, Tianyuan Zou, Eileen Ye, Alphet Liu, Youyong Kong, Ya-Qin Zhang, Duran Zheng, and Maxm Pan. E-Bench: Benchmarking multi-step tool-use agents in real-world product scenarios. *arXiv preprint arXiv:2607.23722*, 2026.

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

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, et al. Qwen-AgentWorld: Language world models for general agents. *arXiv preprint arXiv:2606.24597*, 2026.

A SYNTHESIZED & SAMPLED TASKS

Task Instantiation from Synthesized Worlds. AGENTMERCURY separates world construction from task instantiation. Once an executable world w has been synthesized, multiple tasks can be instantiated from the same underlying world by varying the seeded state, user intent, and task rubric. In our released RL corpus, each task corresponds to an autonomous investigation instance associated with a synthetic company environment and a grounded persona. Thus, the task distribution is sampled on top of a shared distribution of executable worlds rather than being used as the primary specification for constructing those worlds.

Formally, given a synthesized world $w = \langle \mathcal{S}, \mathcal{A}, \Omega, T, O, s_0, \mathcal{R} \rangle$, we instantiate a task as

$$(\Delta s_0, u, \rho) \sim \mathrm{Task}(\cdot \mid s_0, \mathcal{R}), \tag{7}$$

where Δs_0 specifies task-specific state seeding, u is the natural-language user instruction, and ρ is the task-level rubric. The resulting task therefore modifies the initial state and specifies the objective and evaluation criteria without changing the underlying world structure. This separation allows the same world to support multiple task instances and interaction trajectories.

Corpus Composition. In Figure 4, we depict overall statistics of our corpus composition. The RL training corpus used in our experiments contains $43{,}300$ synthesized tasks collected from $4{,}326$ executable company environments, the trainable subset of the $4{,}783$ synthesized environments; an environment is trainable if it passes both task generation and verification. Each trainable environment contributes ten sampled task seeds. The environments span $2{,}287$ distinct industry descriptions and expose 842 unique tools over 222 state-table types and 148 distinct service combinations. Each