Do Enterprise Systems Need Learned World Models? The Importance of Context to Infer Dynamics

Jishnu Sethumadhavan Nair[1,*], **Patrice Bechard**[1,*], **Rishabh Maheshwary**[1,*], **Surajit Dasgupta**[1,†], **Sravan Ramachandran**[1,†], **Aakash Bhagat**[1,†], **Shruthan Radhakrishna**[1,†], **Pulkit Pattnaik**[1], **Johan Obando-Ceron**[2], **Shiva Krishna Reddy Malay**[1], **Sagar Davasam**[1], **Seganrasan Subramanian**[1], **Vipul Mittal**[1], **Sridhar Krishna Nemala**[1], **Christopher Pal**[1,2], **Srinivas Sunkara**[1], **and Sai Rajeswar**[1,2]

[1]ServiceNow Research [2]Mila
[*]Equal contribution. [†]Core contributors.

Abstract

World models enable agents to anticipate the effects of their actions by internalizing environment dynamics. In enterprise systems, however, these dynamics are often defined by tenant-specific business logic that varies across deployments and evolves over time, making models trained on historical transitions brittle under deployment shift. We ask a question the world-models literature has not addressed: *when the rules can be read at inference time, does an agent still need to learn them?* We argue, and demonstrate empirically, that in settings where transition dynamics are configurable and readable, runtime discovery complements offline training by grounding predictions in the active system instance. We propose *enterprise discovery agents*, which recover relevant transition dynamics at runtime by reading the system's configuration rather than relying solely on internalized representations. We introduce *CascadeBench*, a reasoning-focused benchmark for enterprise cascade prediction that adopts the evaluation methodology of World of Workflows on diverse synthetic environments, and use it together with deployment-shift evaluation to show that offline-trained world models can perform well in-distribution but degrade as dynamics change, whereas discovery-based agents are more robust under shift by grounding their predictions in the current instance. The benchmark also isolates a finding that does not follow from rule access alone: even when the active rules are supplied in context, models miss much of the cascade, and recall falls off after the first few rules fire. Rules are necessary but not sufficient; the binding constraint is multi-step rule composition rather than retrieval. Our findings suggest that, in configurable enterprise environments, agents should not rely solely on fixed internalized dynamics, but should incorporate mechanisms for discovering relevant transition logic at runtime, and closing the remaining gap requires learning to compose retrieved rules rather than only to fetch them.

1 Introduction

Large Language Model (LLM) agents [Yao et al., 2022, Wang et al., 2024b] are increasingly deployed in environments with complex dynamics. To plan and act effectively over long horizons, these agents must understand how their actions affect the environment, enabling accurate anticipation of downstream state changes [Erdogan et al., 2025, Gu et al., 2025]. This ability to capture environment dynamics, whether implicitly or explicitly, is central to building reliable autonomous agents in enterprise settings [Gupta et al., 2026].

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

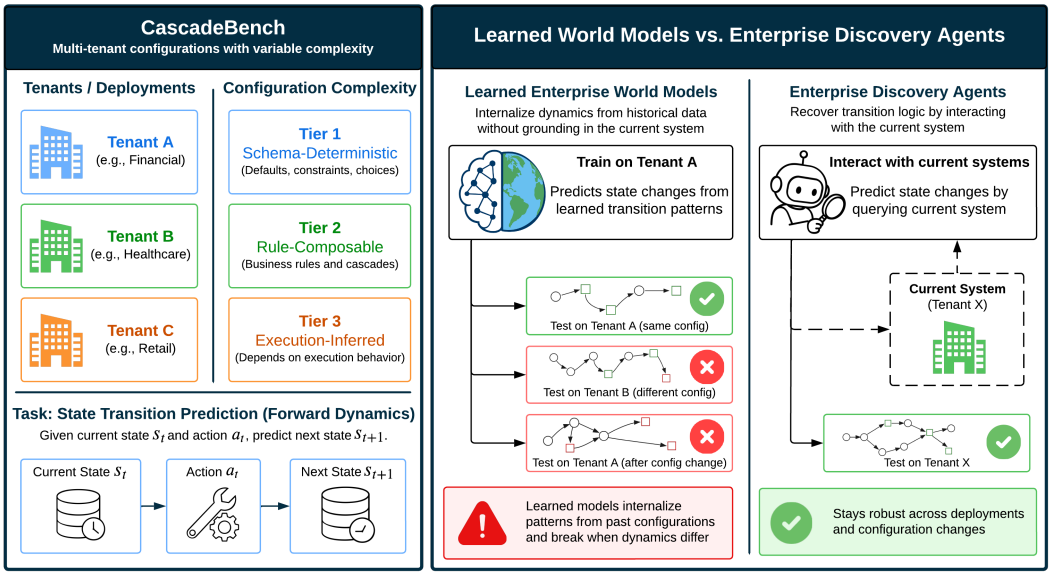

Figure 1: **Overview of CascadeBench and comparison between learned world models and enterprise discovery agents.** *Left*: CascadeBench evaluates agents across multi-tenant enterprise environments and varying configuration complexity tiers. Task requires predicting the next state s_{t+1} given the current state s_t and action a_t. *Right*: Learned enterprise world models internalize transition dynamics from historical data, performing well in-distribution but degrading under deployment and configuration shift when not grounded in the active instance. Enterprise discovery agents interact with the current system (e.g., querying state and inspecting workflow definitions) to recover transition logic at runtime, enabling robust predictions across tenants and evolving configurations.

Enterprise systems differ from traditional environments because their dynamics are partly specified by tenant-specific configuration artifacts, such as business rules and workflows, that vary across deployments and evolve over time [Bezemer and Zaidman, 2010, Makki et al., 2018]. Thus, the same action can have different effects depending on the active configuration of the current system instance. Learned *enterprise world models* can capture recurring patterns within a fixed deployment or workflow family, but models trained only on historical transitions may become brittle under deployment shift [Doshi-Velez and Konidaris, 2016, Lee et al., 2020].

This raises an alternative: instead of internalizing dynamics ahead of time, agents can *discover* them at runtime. We define *enterprise discovery agents* as agents that actively recover transition logic by interacting with the system (e.g. by querying state, inspecting workflow definitions, or issuing targeted probe actions). This strategy is natural in enterprise systems, where transition logic is often exposed through configuration artifacts such as business rules and workflows. Our comparison asks whether agents should rely solely on internalized dynamics when the rules governing the current environment can be inspected directly.

We evaluate this question on *CascadeBench*, a benchmark for enterprise cascade prediction under configuration and deployment shift. We show that offline-trained world models perform well in-distribution but degrade as dynamics change, while discovery agents remain more robust by grounding predictions in the active deployment. These results suggest that enterprise world modeling should combine learned priors with runtime discovery rather than rely only on fixed internalized dynamics.

Our contributions are as follows:

1. We formalize enterprise dynamics as deployment-specific and evolving transition functions defined by system configuration.
2. We introduce *CascadeBench*, a benchmark for evaluating robustness under configuration and deployment shift across varying levels of transition complexity.
3. We propose *enterprise discovery agents*, which recover dynamics at runtime through interaction with the system, improving robustness under shift.

4. We show that offline-trained enterprise world models perform well in-distribution but degrade under configuration shift when not grounded in the active deployment.
5. We show that rule access is necessary but not sufficient: under oracle rule-in-context prompting, models still fail on deep composed cascades, identifying multi-step rule composition rather than retrieval as the binding constraint.

CascadeBench is publicly available at `https://huggingface.co/datasets/ServiceNow-AI/cascade_bench` under the Apache-2.0 license. The discovery agent is implemented in SyGra (`https://github.com/ServiceNow/SyGra/tree/scratch/ewm/tasks/examples/wow_state_predictor_da`).

2 Related Work

World models for decision-making agents. World models aim to enable agents to anticipate the effects of their actions by learning environment dynamics [Hafner et al., 2019, 2020, Hansen et al., 2024]. Early work by Schmidhuber [1990] introduced the idea of separating a predictive model and control, forming the foundation of model-based reinforcement learning. This paradigm has since been extended by methods such as World Models [Ha and Schmidhuber, 2018] and Dreamer [Hafner et al., 2020, 2025], which learn latent dynamics to support planning and policy optimization. Later work improves scalability through better latent representations, longer-horizon rollouts, and tighter integration between planning and learning [Hansen et al., 2024]. In visual and robotic settings, approaches such as the I-JEPA [Assran et al., 2023] and V-JEPA [Assran et al., 2025] similarly motivate learning predictive representations rather than predicting pixels directly.

More recently, world models have been adapted to language-based agents, where reasoning is framed as planning over simulated trajectories [Hao et al., 2023]. In these settings, the environment is a structured interface such as the web, code execution environments, or tool APIs. Methods such as WebDreamer [Gu et al., 2025], Code World Models [Copet et al., 2025], and Generative Tool Models (GTM) [Ren et al., 2025] learn to approximate environment responses, enabling agents to simulate interactions without executing them. Across these approaches, the common assumption is that environment dynamics should be internalized into a learned simulator. We study a complementary regime where system behavior is externally accessible at inference time through structured interfaces, logs, or configuration files. In such settings, learned simulators may introduce unnecessary approximation error and reduce robustness under distribution shift.

Agents interacting with structured environments. A complementary line of work studies agents that interact directly with environments to retrieve information or execute actions. Tool-augmented agents [Yao et al., 2022, Schick et al., 2023] use external APIs and structured interfaces to ground reasoning in real system responses. Recent work shows that such agents can operate effectively in enterprise environments by querying platform APIs at runtime, avoiding the need to approximate system behavior [Bechard et al., 2026]. Beyond tool use, interaction can also serve as a mechanism for structure discovery. Agents can acquire reusable skills through exploration [Wang et al., 2024a], infer abstractions from structured interfaces [Prabhu et al., 2026], and recover latent environment dynamics through experimentation [Jansen et al., 2024]. These approaches suggest that interaction provides a reliable and adaptive signal for understanding environment behavior, particularly in non-stationary or partially observable settings. Our work builds on this perspective by studying agents that explicitly recover transition dynamics from live system configurations, enabling robust behavior under distribution shift rather than relying solely on learned simulators.

Enterprise agent benchmarks. Existing enterprise benchmarks evaluate agents on task execution across UI- and API-based settings. UI-centric benchmarks such as WorkArena and WorkArena++ [Drouin et al., 2024, Boisvert et al., 2024] focus on browser interaction with platforms like ServiceNow, exposing challenges in long-horizon planning, delayed feedback, and error accumulation. API-based benchmarks such as CRMArena [Huang et al., 2025] operate over structured Salesforce environments, enabling more controlled evaluation but often restricting the action space and system complexity. Multi-domain settings like EnterpriseOps-Gym [Malay et al., 2026] and TheAgentCompany [Xu et al., 2026] expand coverage across enterprise tools and workflows, though they primarily emphasize task execution rather than understanding system dynamics.

World of Workflows (WoW) [Gupta et al., 2026] takes a different angle, evaluating agents' ability to predict state transitions, action effects, and constraints in enterprise workflows, showing that frontier models struggle with multi-step dynamics. However, WoW evaluates fixed configurations in zero-shot settings, leaving open how agents adapt when dynamics vary across deployments. We address this gap with CascadeBench, a reasoning-focused benchmark that adopts WoW's transition-prediction methodology on synthetic schemas designed to isolate reasoning from parametric memorization and retrieval noise. Rather than measuring prediction accuracy under fixed configurations, we study how agents recover and adapt to dynamics at inference time.

3 Enterprise Dynamics

An enterprise platform typically maintains a structured state encoded across interconnected database tables, which may include information such as: users, configuration items, incidents, changes, and Service Level Agreements (SLAs). An agent interacts with this state through API actions to create records, update fields, trigger workflows, etc. The consequences of any action depend not only on the current state and the action itself, but on a layer of customer-specific configuration that governs how the platform responds. We formalize this as a contextual transition model. Let s_t denote the observable platform state at step t (the set of record field values across all relevant tables), a_t the action taken, and c the instance configuration (the collection of all business rules[1], workflow definitions, approval policies, SLA definitions, and access control lists deployed on a particular customer's instance),

$$s_{t+1} \sim P(s_{t+1} \mid s_t, a_t, c). \tag{1}$$

In standard world model settings, c is fixed and unknown, and the agent must learn dynamics from interaction alone. Enterprise systems differ from standard world model settings in two ways. First, c is not fixed. Administrators continuously modify rules, so dynamics shift without changes to the underlying platform. Second, c is explicit and readable. Rules, workflows, and policies are stored as inspectable records with defined conditions and actions. The central question is whether a learned world model trained on transition data can reliably predict s_{t+1} on its own, or whether accurate prediction requires runtime grounding in the active configuration c. Furthermore, unlike formulations that model the full environment state, enterprise world models benefit from a sparse transition view. In practice, we effectively model a state delta, Δs_t, roughly corresponding to $s_{t+1} - s_t$: the subset of fields whose values change after action a_t. This focuses modeling capacity on the task-relevant parts of the enterprise state affected by the transition.

Actions compose through cascades: a single field update can induce chains of business rule executions that propagate across tables, initiate SLA timers, and schedule notifications. The resulting transition from s_t to s_{t+1} may therefore involve dozens of intermediate steps, with depth and branching determined entirely by the instance-specific configuration of interacting rules.

Not all state transitions are equally hard to predict. To clarify the sources of difficulty in this setting, we distinguish three levels of transition complexity: *Tier 1* schema-determined effects, *Tier 2* rule-composed cascades, and *Tier 3* execution-inferred behavior. Table 5 in the Appendix summarizes this taxonomy with concrete examples. We use these tiers both to structure the benchmark and to scope our comparison. Tier 1 and Tier 2 transitions are recoverable, in principle, from inspectable configuration: schemas capture defaults and constraints, while active business rules capture multi-step cascades. Tier 3 transitions are only partially recoverable: the rules are still inspectable, but the realized outcome also depends on execution-order resolution and other engine-internal behaviors not exposed in static artifacts. We therefore treat Tier 3 as a partial structural limit rather than a hard ceiling, and report tier-stratified results separately.

4 Enterprise Gym

We define a world as $W = (E, T)$, where E specifies the environment (organizational structure, configuration database, business rules, initial records, etc.) and T is the transition function induced by E on the platform. T is not simulated: we deploy E to a live platform instance so that when an agent acts, the real engine executes server-side scripts and the resulting state s' is the actual database state, avoiding the simulation-to-production gap of approximated benchmarks.

[1] Abbreviated BR throughout. See Appendix J for ServiceNow-specific terminology used in this paper.

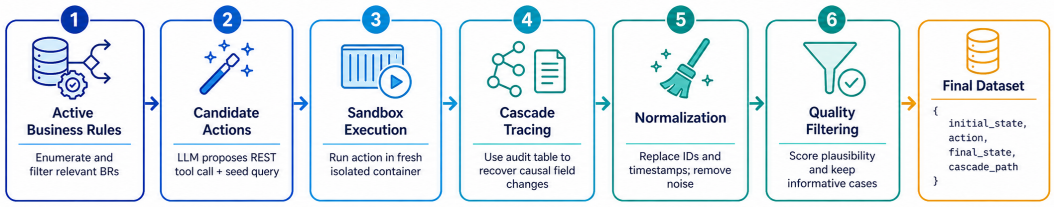

Figure 2: **Pipeline for constructing the Business Rule cascade dataset.** Candidate actions are executed in isolated sandboxes, traced through audit logs, normalized, and filtered into a dataset of state transitions and cascade paths.

Diversity at scale. Worlds are generated from a catalog of 1,596 business rule patterns spanning 6 industries and 11 operational domains, with each world instantiating a unique subset. A dependency-ordered construction pipeline expands ∼27,000 LLM-generated base scenarios into ∼802,000 validated initial states. Diversity mechanisms, controlled rule-conflict injection, and validation guardrails are described in Appendix E.

Data Collection. We collect ground-truth transition data by firing tool calls against the live worlds described above and recording the resulting cascades through platform audit logs. Figure 2 summarizes the pipeline: candidate tool calls are executed in isolated sandboxes, causal state changes are recovered from `sys_audit`, platform-specific identifiers and noise are normalized away, and low-quality traces are filtered before inclusion. Each retained sample is a tuple (s_t, a_t, s_{t+1}, π), where s_t is the relevant initial state, a_t is the executed tool call, s_{t+1} is the post-execution state diff, and π is the cascade path: the ordered sequence of tables touched and business rules attributed to each transition. The platform engine is the source of truth: when an action is fired, real business rules execute, real SLA timers start, and real cascades propagate. We do not simulate any part of T.

The resulting corpus contains 27,243 verified transition samples spanning 64 worlds across 6 industries (financial services, government, healthcare, manufacturing, retail, technology) and 3 organizational sizes (small, midmarket, enterprise). We construct train/test splits at the world level, stratified by industry and organizational size, and reserve held-out industry–size combinations for evaluation. As a result, evaluation requires generalizing to unseen deployment regimes rather than interpolating among samples from the same worlds.

Benchmarking. We construct *CascadeBench*, to evaluate transition prediction under controlled configuration shift. CascadeBench retains WoW's evaluation methodology: models predict field-level state changes from a proposed action, and predictions are compared against audit-log ground truth. However, CascadeBench is designed to isolate *reasoning* over provided rules from confounding factors present in existing benchmarks, including parametric memorization, retrieval noise, and audit-log artifacts.

Concretely, CascadeBench differs from existing enterprise benchmarks along three axes. First, it is built on synthetic schemas that do not appear in real platform deployments, so models cannot rely on memorized table structures. Second, CascadeBench makes the relevant context available for each example—table schemas, business rules, and seed records—so we can control how much context the model receives. This enables both fully contextualized evaluation and context-limited settings that probe internalized knowledge or the effectiveness of runtime discovery. Third, audit-log ground truth is restricted to content fields, removing engine-internal metadata that does not reflect business logic, such as system identifiers, timestamps, and bookkeeping fields. Together, these choices let CascadeBench disentangle memorization, context discovery, and rule-based reasoning. CascadeBench contains 739 validated transitions from 37 trajectories (Tier 1: 177, Tier 2: 424, Tier 3: 138). We describe the construction pipeline in Appendix D. We provide a comparison between CascadeBench and WoW in Appendix F.

What is real and what is synthetic. CascadeBench is synthetic in its surface form but real in its dynamics. Every transition is executed by the platform's own rule engine and read back from its audit log, so nothing is simulated. The schemas, on the other hand, are generated fresh, so that models cannot rely on tables they may have seen during pretraining. The rules and cascades are modelled on documented platform patterns rather than written from scratch. Appendix D.1 explains each of these choices in more detail.

5 Approaches

We compare three approaches to predict enterprise state transitions: prompting a frozen model, fine-tuning a learned world model, and using a discovery agent that inspects the current instance at inference time. All approaches take a current state s_t and proposed action a_t as input and output the same target representation: structured field-level diffs describing the predicted transition.

5.1 Prompted Baseline

The prompted baseline uses a frozen language model to predict the effects of an action from the provided context alone. Given s_t and a_t, the model outputs the expected state change as a structured set of field-level diffs. This baseline measures how well general-purpose models can infer transition behavior without fine-tuning or runtime access to instance-specific configuration. Depending on the evaluation setting, the prompt may include only the action and relevant state, or additional provided context such as schemas and rules.

5.2 Learned Enterprise World Model

The learned enterprise world model predicts transitions by internalizing dynamics from supervised data. We fine-tune on (s_t, a_t, s_{t+1}) tuples collected from Enterprise Gym (§4), with the target being the minimal field-level diff between s_t and s_{t+1}. This tests whether learned dynamics transfer to instances with different industries, organizational structures, and rule sets.

5.3 Enterprise Discovery Agent

The enterprise discovery agent predicts the outcome of a proposed action without executing it and without updating model parameters. Unlike learned world models, it does not attempt to internalize environment dynamics. Instead, it queries the live instance configuration c and reasons over the retrieved information to infer the effects of an action.

We model enterprise transitions as depending on instance-specific configuration:

$$s_{t+1} \sim P(s_{t+1} \mid s_t, a_t, c), \tag{2}$$

where c denotes the deployed configuration of the current instance. Since c can be large, the discovery agent follows a retrieve-then-reason strategy. Given s_t and a_t, it retrieves a task-relevant subset $\tilde{c} \subseteq c$ and predicts the next state as

$$\hat{s}_{t+1} = f_{\text{LLM}}(s_t,\ a_t,\ \tilde{c},\ \hat{s}_{1:t})\,, \tag{3}$$

where f_{LLM} is a frozen language model and $\hat{s}_{1:t}$ denotes prior predictions in a multi-step rollout.

Retrieval is adaptive: simple transitions may require little or no additional context, while more complex cascades trigger targeted queries for relevant rules, schemas, records, or SLA definitions. For multi-step rollouts, predictions are generated sequentially, with each $\hat{s}_i$ appended to the context before predicting $\hat{s}_{i+1}$, enabling the agent to reason about compounding effects across the cascade chain (§7). Because $\tilde{c}$ is retrieved at inference time rather than memorized during training, the same agent transfers across tenants of the same enterprise platform without modification. To isolate the contribution of runtime discovery, the static context is matched to that of prompted baselines; any improvement can therefore be attributed to retrieval and reasoning over $\tilde{c}$. Architecturally, the agent is a standard ReAct-style tool-use loop [Yao et al., 2022]; we do not propose a new retrieval or planning algorithm. We keep it deliberately simple so that any gain over the matched prompted baseline can be attributed to grounding in the live configuration rather than to the agent design. Implementation details for the enterprise discovery agent can be found in Appendix G.

6 Experiments

We organize the analysis as a three-rung ladder. Each rung is a declarative claim about where the dynamics come from at prediction time, with evidence drawn from Table 1. The ladder is deliberately ordered so that the expected result sets up the unexpected one. Rungs 1 and 2 establish that rules are load-bearing and that internalization binds to the training distribution, neither of which is surprising. Rung 3 and the tier analysis then show that supplying the rules, by prompt or by retrieval, does not close the gap: composition is what remains.

Table 1: **Main transition-prediction results on CascadeBench and WoW.** We report IoU and table/field-level IoU (IoU(T+F)) with and without access to business rules (BR). Bold indicates the best score in each column for each model type.

		CascadeBench				WoW	
		w/ BR		w/o BR		w/o BR	
Type	**Model**	**IoU**	**IoU(T+F)**	**IoU**	**IoU(T+F)**	**IoU**	**IoU(T+F)**
Frontier Models	Sonnet 4.6 [Anthropic, 2026b]	38.15	57.67	10.30	10.53	38.65	41.54
	Opus 4.6[Anthropic, 2026a]	40.46	59.69	**10.91**	**16.15**	**41.32**	**44.90**
	GPT-5.2[OpenAI, 2025]	38.40	59.24	9.77	13.18	23.97	26.95
	GPT-5[Singh et al., 2025]	**41.78**	**61.50**	9.82	13.25	29.34	32.32
	Gemini 3 Pro[Gemini Team, 2025]	41.36	59.53	10.38	13.92	35.59	39.06
Base Models	Qwen-3.5-27B[Qwen Team, 2026a]	**40.39**	**60.86**	9.77	13.18	21.22	15.19
	Qwen-3.6-27B[Qwen Team, 2026b]	38.98	60.06	7.13	13.15	**23.17**	**27.02**
	Gemma-4-31B[Google DeepMind, 2025]	37.99	60.32	**9.88**	**13.76**	21.11	23.55
Finetuned Models	Qwen-3.5-27B-LoRA	**50.90**	**61.47**	10.60	14.51	31.21	35.02
	Qwen-3.6-27B-LoRA	40.47	60.93	9.74	13.40	**32.22**	**36.88**
	Gemma-4-31B-LoRA	41.33	52.62	**12.19**	**16.41**	31.73	36.07

Models. We fine-tune Qwen-3.5-27B [Qwen Team, 2026a], Qwen-3.6-27B [Qwen Team, 2026b], and Gemma-4-31B-it [Google DeepMind, 2025] with LoRA [Hu et al., 2022] on the transition tuples from §4. The same models are evaluated zero-shot as prompted baselines, together with frontier models (Claude Sonnet 4.6 [Anthropic, 2026b], Claude Opus 4.6 [Anthropic, 2026a], GPT-5 [Singh et al., 2025], GPT-5.2 [OpenAI, 2025], Gemini 3 Pro [Gemini Team, 2025]).

Metrics. All methods take (s_t, a_t) as input and predict field-level diffs, which we score against audit-log ground truth using two complementary IoU variants from Gupta et al. [2026]. **IoU(T+F)** credits a prediction when it correctly identifies the affected *(table, field)* pair, capturing whether the model has identified *what changes* in the global state. Strict **IoU** additionally requires the predicted value to match, capturing *how it changes*. We report both because identifying which elements of the global state will be impacted is itself a substantial part of the prediction problem in enterprise environments—a state diff over a database with thousands of fields requires the model to first localize the cascade footprint before reasoning about specific values.

Evaluation settings. On *CascadeBench* we run two settings: **w/ BR** supplies the relevant business rules in the prompt (an oracle for retrieval), and **w/o BR** removes them (testing what the model knows on its own). These settings separate the factors a reader might expect to drive performance. Unseen synthetic schemas are a memorization control, deployment shift is the setting under study, and the w/ BR oracle removes retrieval entirely; what the oracle cannot remove is the need to compose the supplied rules across a multi-hop cascade, which is the capability CascadeBench is designed to isolate. We also report results on the WoW benchmark, which runs the same prediction task on real ServiceNow instances with no business rules in the prompt. Because WoW is constructed independently of our data pipeline, it also serves as a transfer test: models fine-tuned only on Enterprise Gym transitions can be compared with their base versions on WoW's next-state prediction task. To our knowledge, WoW and CascadeBench are the only benchmarks that score next-state prediction on a live enterprise instance; task-completion benchmarks such as WorkArena [Drouin et al., 2024], CRMArena [Huang et al., 2025], and EnterpriseOps-Gym [Malay et al., 2026] do not expose the intermediate cascades this requires.

Rung 1: Prompting alone struggles when rules are hidden; SFT helps mainly without rule context. Table 1 shows that when business rules are not provided, prompted models perform poorly on CascadeBench, with both frontier and base open-weight models in the 9–16 IoU(T+F) range. This is substantially lower than the 21–23 range observed for base models on WoW, suggesting that CascadeBench poses a harder transition-prediction problem with more hidden or cascading dynamics. SFT on the transition tuples from §4 improves this no-BR setting, yielding modest gains on CascadeBench w/o BR ($\sim$2–3 IoU points) and larger gains on WoW ($\sim$10 points). With business rules in context, however, SFT is not uniformly beneficial: Qwen-3.5-27B reaches 50.9 IoU, above the 38–42 range of prompted models with BRs, but other models gain little or regress. Thus, SFT helps when rule context is missing, but does not consistently substitute for grounding in the active rules.

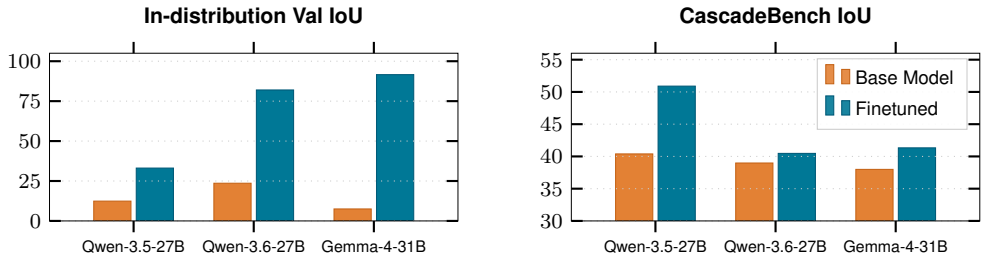

Figure 3: In-distribution Val IoU & CascadeBench IoU (out-of-distribution) comparing base models and finetuned counterparts. All settings have access to business rules.

Table 2: **Discovery agent vs. trained world model on CascadeBench.** *Oracle* provides business rules in context (reasoning ceiling). *DA* starts without rules recovers them via retrieval at inference. *Trained:* a LoRA-finetuned model evaluated without rules (internalized dynamics). *Prompted* is the no-context, no-training floor. On models where both DA and Trained are available, DA substantially exceeds Trained on same models.

Type	Model	Oracle w/ BR	DA w/o BR	Prompted w/o BR
Frontier Models	GPT-5 [Singh et al., 2025]	41.78	32.1	9.82
	Claude Opus 4.6 [Anthropic, 2026a]	40.46	32.0	10.91
	Claude Sonnet 4.6 [Anthropic, 2026b]	38.15	29.7	10.30
	Gemini 3 Pro [Gemini Team, 2025]	41.36	31.2	10.38
	GPT-5.2 [OpenAI, 2025]	38.40	24.9	9.77
Finetuned Models	Qwen-3.6-27B-LoRA [Qwen Team, 2026b]	40.47	**28.8**	9.74
	Qwen-3.5-27B-LoRA [Qwen Team, 2026a]	50.90	**21.5**	10.60
	Gemma-4-31B-LoRA [Google DeepMind, 2025]	41.33	12.5	12.19

Rung 2: SFT is strong in distribution but degrades under shift. Figure 3 shows that fine-tuning can strongly internalize the training dynamics: validation IoU rises to 91.6 for Gemma-4-31B and 82.0 for Qwen-3.6-27B, far above their base counterparts. However, this advantage largely collapses on CascadeBench, where models face synthetic schemas and configurations not seen during training: both models fall to roughly 40–41 IoU. Fine-tuned models remain stronger than prompted baselines, but most of their in-distribution edge is lost under shift. This suggests that SFT learns useful transition patterns, but also binds them to the training distribution; internalization alone is therefore insufficient for robust cross-instance prediction.

Rung 3: Runtime discovery recovers cross-instance accuracy. If neither the prompt nor the weights solve the problem on their own, the remaining option is to recover the rules from the live instance at inference time. The discovery agent (§5.3) uses the same static context as the prompted baseline, but can additionally query the live instance for rules, schemas, and records before predicting. Figure 4 shows state-prediction IoU on WoW across rollout horizons $k{=}1, \ldots, 5$. Discovery improves over the matched prompted baseline for every model and horizon we evaluate, including at $k{=}1$, where Opus 4.6 rises from 0.40 to 0.45 and GPT-5 Mini from 0.20 to 0.26. The margin varies across settings: discovery and prompting are sometimes close, but discovery also yields substantial gains of up to roughly 0.10 IoU. Importantly, the advantage remains visible across rollout depths despite compounding errors. This suggests that querying the live instance provides complementary signal beyond the static prompt, improving cross-instance prediction without training on the target instance.

Retrieval beats internalization on the same model. Table 2 compares three ways of accessing transition logic on CascadeBench: providing business rules in context, retrieving them at inference time, or relying on the prompt/weights without retrieval. For the non-finetuned frontier models, the static no-BR baseline is consistently low around 10 IoU, while the discovery agent recovers a large fraction of the oracle signal, reaching the mid-20s to low-30s without any training on the target instance. This shows that the benefit of discovery is not specific to fine-tuned models. On the LoRA models, the same-model comparison is more nuanced: retrieval substantially outperforms internalization for the Qwen models, while Gemma is roughly tied and remains far below the oracle setting. Overall, runtime retrieval is a more reliable source of cross-instance signal than static prompting or weights alone, but the remaining gap to the oracle indicates that retrieval and rule

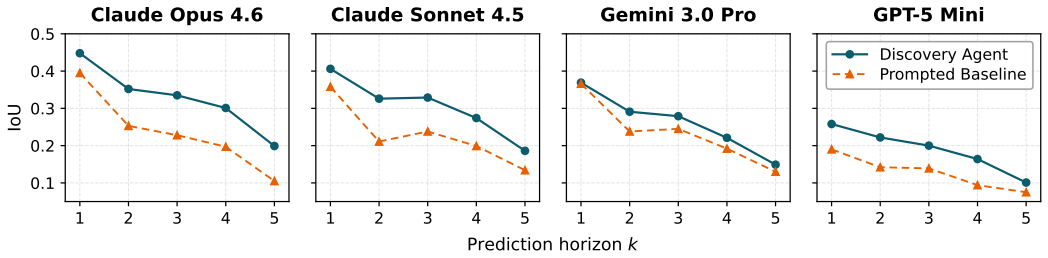

Figure 4: **Runtime discovery improves multi-step state prediction.** IoU across prediction horizons $k = 1, \ldots, 5$ for four backbone models. Performance degrades as the rollout horizon increases, but the Discovery Agent remains consistently above the prompted baseline, indicating that retrieving transition logic at inference time helps reduce compounding prediction errors.

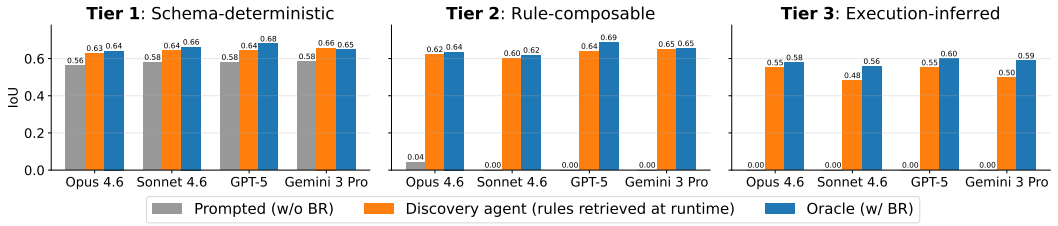

Figure 5: **Tier-stratified transition prediction on CascadeBench.** Prompted models are often sufficient for simple schema-determined effects, but fail on harder tiers where hidden workflows, rule cascades, and execution-dependent effects shape the next state. Runtime discovery recovers much of the information available to the oracle rule-in-context setup.

composition are still imperfect. This motivates training discovery agents to retrieve and compose the active rules more effectively.

Discovery helps most when transition logic exceeds the schema. Figure 5 stratifies CascadeBench by transition complexity for proprietary models, following §3. Prompting without rules handles Tier 1 schema effects reasonably well, reaching roughly 0.56–0.58 IoU, but nearly collapses on Tier 2 cascades and Tier 3 conflicts because rule context is missing. Runtime discovery recovers most of this gap: across Claude Opus 4.6, Claude Sonnet 4.6, GPT-5, and Gemini 3 Pro, it stays near the oracle on Tier 1 and Tier 2 while outperforming the prompted baseline. The main remaining gap is Tier 3, where outcomes depend on execution semantics not fully exposed in configuration. Thus, discovery is most valuable where static prompting fails: hidden rules and cascades, not schema-only effects.

Discovery under realistic constraints. In an ablation on World of Workflows (37 trajectories, five-step rollouts, GPT-5.2 [OpenAI, 2025] and Claude Sonnet 5 [Anthropic, 2026c], one seed per model), the discovery agent uses about 13 tool calls per trajectory. Injecting retrieval failures at rates up to 50%, or denying access to all configuration tables, leaves overall IoU essentially unchanged, and the gap between directly edited fields and cascaded fields persists in every condition. Accuracy therefore does not hinge on full queryability, consistent with composition rather than retrieval being the bottleneck.

7 Discussion

Effect of Business Rules. Across the model classes in Table 1, removing business rules from the prompt produces a uniform collapse on CascadeBench. With BR, every class — frontier, base, and SFT — sits in a 38–51 IoU band; without BR, the same models fall to 7–12. The drop is consistent across model sizes and families. This demonstrates that *business rules carry the dynamics that CascadeBench measures*: the benchmark probes rule-grounded reasoning rather than what models already know from pretraining. Furthermore, since rules cannot be replaced by pretraining or fine-tuning, supplying them at inference time in the prompt or via runtime retrieval is the deciding factor for prediction accuracy.

Depth Analysis. Figure 4 shows that Discovery Agents outperform matched prompted baselines across rollout horizons. Performance generally decreases as k grows for both methods, as longer horizons require predicting deeper cascades and create more opportunities for error accumulation. Despite this increasing difficulty, discovery remains consistently beneficial from $k{=}1$ through $k{=}5$. Across all matched models and rollout horizons, the Discovery Agent improves over the prompted baseline, with the size of the gain varying by model and depth.

The likely mechanism is repeated grounding. Both methods condition later predictions on earlier predicted diffs, so neither is immune to compounding errors. However, the prompted baseline relies primarily on its initial context and prior outputs, while the Discovery Agent can re-query the live instance for relevant records, active business rules, and reference identifiers at each step. This refreshes the model's view of the deployed configuration rather than relying only on its evolving prediction state. Thus, the same backbone model produces stronger long-horizon predictions when placed inside a discovery loop, showing that runtime retrieval improves robustness in multi-step cascade prediction.

8 Conclusion

This paper studies transition prediction in enterprise environments, where dynamics are shaped by tenant-specific configurations rather than fixed rules inferred only from experience. We find that offline-trained world models perform well in-distribution but degrade on held-out configurations. Discovery agents, which retrieve relevant rules at inference time, remain more robust under shift and avoid some of the error compounding observed in purely internalized models.

Discovery agents are not a replacement for learned world models. Instead, our results suggest that when transition logic is readable from the live system, agents should not rely solely on internalized dynamics. The next step is to combine learned priors with runtime retrieval and reasoning: training agents that learn when, what, and how to retrieve. We discuss the scope and assumptions of this conclusion in Section 9.

9 Limitations

The discovery agent assumes business rules are readable on the live instance; access controls degenerate it to the prompted baseline. DA performance also depends on tool-use capability: on open-weight models in the 27–31B range, the retrieval loop is unreliable enough that LoRA finetuning wins in some conditions, so the choice between training and discovery is deployment-dependent. Our evaluation is single-platform (ServiceNow) and our quantitative results focus on Tier 1 and Tier 2 transitions; Tier 3 results are reported in Appendix C; Tier 3 stratification is limited to multi-rule conflicts detectable from the audit log, and broader execution-order dynamics remain out of scope. The DA-vs-trained comparison rests on a small set of open-weight models where LoRA finetuning is feasible. When rules are only partially readable, performance should lie between the no-BR floor and the oracle ceiling we report, and we do not measure intermediate visibility directly. Runtime discovery also adds latency and tool-call cost, must operate within query budgets and permission scopes, and inherits a synthetic-to-real gap from CascadeBench's generated schemas. Appendix A expands on each.

10 Acknowledgements and Disclosures

Funding. This work was conducted at ServiceNow Research and Mila. It received no third-party funding.

Competing interests. Authors affiliated with ServiceNow Research are employed by ServiceNow, whose platform is used to run every experiment and on which CascadeBench and Enterprise Gym are built. The authors declare no other competing interests.

References

Anthropic. Claude opus 4.6 system card. Technical report, Anthropic, February 2026a. URL https://www-cdn.anthropic.com/6a5fa276ac68b9aeb0c8b6af5fa36326e0e166dd.pdf. Accessed: 2026-03-13.

Anthropic. Claude sonnet 4.6 system card. Technical report, Anthropic, February 2026b. URL https://www-cdn.anthropic.com/bbd8ef16d70b7a1665f14f306ee88b53f686aa75.pdf. Accessed: 2026-03-13.

Anthropic. Claude sonnet 5 system card. Technical report, Anthropic, June 2026c. URL https://www.anthropic.com/claude-sonnet-5-system-card. Accessed: 2026-11-05.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15619–15629, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Patrice Bechard, Orlando Marquez Ayala, Emily Chen, Jordan Skelton, Sagar Davasam, Srinivas Sunkara, Vikas Yadav, and Sai Rajeswar. Terminal agents suffice for enterprise automation. *arXiv preprint arXiv:2604.00073*, 2026.

Cor-Paul Bezemer and Andy Zaidman. Multi-tenant saas applications: maintenance dream or nightmare? In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*, IWPSE-EVOL '10, page 88–92, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450301282. doi: 10.1145/1862372.1862393. URL https://doi.org/10.1145/1862372.1862393.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier de Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=PCjK8dqrWW.

Jade Copet, Quentin Carbonneaux, Gal Cohen, Jonas Gehring, Jacob Kahn, Jannik Kossen, Felix Kreuk, Emily McMilin, Michel Meyer, Yuxiang Wei, et al. Cwm: An open-weights llm for research on code generation with world models. *arXiv preprint arXiv:2510.02387*, 2025.

Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: a semiparametric regression approach for discovering latent task parametrizations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1432–1440. AAAI Press, 2016. ISBN 9781577357704.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: how capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Lutfi Eren Erdogan, Hiroki Furuta, Sehoon Kim, Nicholas Lee, Suhong Moon, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ybA4EcMmUZ.

Gemini Team. Gemini 3 pro model card. Technical report, Google DeepMind, December 2025. URL https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf. Accessed: 2026-05-04.

Google DeepMind. Gemma model card. https://ai.google.dev/gemma/docs/core/model_card_4, 2025. Accessed: 2026-04-27.

Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. Is your LLM secretly a world model of the internet? model-based planning for web agents. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=c6l7yA0HSq.

Lakshya Gupta, Litao Li, Yizhe Liu, Sriram Ganapathi Subramanian, Kaheer Suleman, Zichen Zhang, Haoye Lu, and Sumit Pasupalak. World of workflows: a benchmark for bringing world models to enterprise systems. *arXiv preprint arXiv:2601.22130*, 2026.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.507. URL https://aclanthology.org/2023.emnlp-main.507/.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Kung-Hsiang Huang, Akshara Prabhakar, Sidharth Dhawan, Yixin Mao, Huan Wang, Silvio Savarese, Caiming Xiong, Philippe Laban, and Chien-Sheng Wu. CRMArena: Understanding the capacity of LLM agents to perform professional CRM tasks in realistic environments. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.

Peter Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi Mishra, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. Discoveryworld: A virtual environment for developing and evaluating automated scientific discovery agents. *Advances in Neural Information Processing Systems*, 37:10088–10116, 2024.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Majid Makki, Dimitri Van Landuyt, Bert Lagaisse, and Wouter Joosen. A comparative study of workflow customization strategies: Quality implications for multi-tenant saas. *Journal of Systems and Software*, 144:423–438, 2018. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2018.07.014. URL https://www.sciencedirect.com/science/article/pii/S0164121218301420.

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.

OpenAI. Update to gpt-5 system card: Gpt-5.2. Technical report, OpenAI, December 2025. URL https://cdn.openai.com/pdf/3a4153c8-c748-4b71-8e31-aecbde944f8d/oai_5_2_system-card.pdf. Accessed: 2026-03-13.

Viraj Prabhu, Yutong Dai, Matthew Fernandez, Krithika Ramakrishnan, Jing Gu, Yanqi Luo, silvio savarese, Caiming Xiong, Junnan Li, Zeyuan Chen, and Ran Xu. WALT: Web agents that learn tools. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=cgIDqcJcoI.

Bidyapati Pradhan, Surajit Dasgupta, Amit Kumar Saha, Omkar Anustoop, Sriram Puttagunta, Vipul Mittal, and Gopal Sarda. Sygra: A unified graph-based framework for scalable generation, quality tagging, and management of synthetic data. *arXiv preprint arXiv:2508.15432*, 2025.

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

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-27b.

Zhenzhen Ren, Xinpeng Zhang, Zhenxing Qian, Yan Gao, Yu Shi, Shuxin Zheng, and Jiyan He. GTM: Simulating the world of tools for AI agents, 2025. URL https://arxiv.org/abs/2512.04535.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Jürgen Schmidhuber. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments, 1990. URL https://people.idsia.ch/~juergen/FKI-126-90ocr.pdf. Technical Report FKI-126-90, Technische Universität München.

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

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024b.

Frank F. Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Zhiruo Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Melroy Maben, Raj Mehta, Wayne Chi, Lawrence Keunho Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. Theagentcompany: Benchmarking LLM agents on consequential real world tasks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=LZnKNApvhG.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Appendix

Table of Contents

Page

A. Limitations 15
B. Discovery Agent Scores 16
C. Tier-Stratified Results 16
D. CascadeBench: Construction Pipeline 17
D.1 What is real and what is synthetic 18
E. Enterprise Gym: World Construction Details 19
F. WoW vs CascadeBench 20
G. Discovery Agent Implementation 20
H. CascadeBench: Failure Mode Analysis 22
H.1 Aggregate Failure Analysis 23
I. Fine-tuning details 24
J. Glossary 24

A Limitations

Inspectability assumption. The discovery agent assumes the relevant business rules and supporting tables are readable on the live instance. Production deployments often impose access controls, in which case runtime discovery degenerates to the prompted baseline.

Partial rule visibility. Our evaluation measures the two ends of the visibility range: no rule access (w/o BR) and full access (the w/ BR oracle), both in Table 1. A deployment where only some rules are readable should fall between them. The tier analysis (Figure 5) suggests where: schema-determined Tier 1 effects are predictable without rules, so reduced visibility mainly affects the rule-composable and execution-inferred tiers. We do not evaluate partial visibility levels directly.

Deployment cost and constraints. Runtime discovery adds latency and tool-call cost to each prediction; our agent is allowed up to 15 recursive tool calls (Appendix G), and we do not analyze sensitivity to tighter query budgets here. Production instances may also impose rate limits, security and privacy restrictions on which tables an agent may read, and incomplete permissions, each of which narrows the context available to discovery.

Synthetic-to-real gap. CascadeBench transitions are computed by the real platform engine, but its schemas are synthetic and its rule structure is abstracted from documented platform patterns rather than taken from customer deployments. Results may therefore not carry over exactly to production tenants, whose rule sets are larger and less regular; the WoW results on real ServiceNow instances provide a partial check.

Tool-use capability bounds discovery. DA performance depends on the model's ability to issue and reason over tool calls. On open-weight models in the 27–31B range, the retrieval loop is unreliable: on Gemma-4-31B-LoRA, the discovery agent reaches 12.5 IoU, far below the 41.33 of the same model given the rules, and no better than its 12.19 without retrieval (Table 2). The right choice between training and discovery is therefore deployment-dependent: frontier APIs favor discovery, constrained open-weight deployments favor finetuning.

Single-platform evaluation. Our experiments evaluate on ServiceNow. Other enterprise platforms have different rule formalisms, cascade semantics, and inspectability guarantees. We expect the inversion argument to transfer (configurability is a property of all major enterprise platforms) but do not demonstrate it directly. The mechanism we rely on requires only that transition dynamics come from inspectable, executable configuration and that the true next state can be read back from the engine, which holds for the major enterprise platforms rather than for IT service management alone. Cross-platform validation is left to future work.

Tier 3 dynamics. Tier-stratified results, including a Tier 3 stratum, are reported in Appendix C. Our Tier 3 operationalization is restricted to multi-rule conflicts where two or more rules write distinct values to the same field, which are detectable from the audit log. Broader execution-inferred dynamics, such as async/sync interleaving, race conditions across parallel rule firings, and platform-internal scheduling behaviors, are present in the Enterprise Gym corpus by construction but are not separately stratified, since attributing them at scale requires resolving execution-order semantics that the audit log alone does not expose. Extending the Tier 3 stratum to cover these cases is left to follow-up work.

Same-model comparison scope. The DA-vs-trained comparison in Table 2 is restricted to models where LoRA finetuning is feasible (Qwen-3.5/3.6-27B, Gemma-4-31B). The finding that retrieval beats internalization on a fixed model therefore rests on a small set of open-weight models.

Amortizing discovery. Discovery need not run in full at every prediction. Because a process's side effects are largely a property of its configuration rather than of any single query, a discovery pass could be run once offline to record a process's expected cascades in natural language (a reusable "skill") for later predictions to read, with re-discovery triggered only for processes whose configuration changes. We leave this direction to future work.

Table 3: Cascade prediction performance on WoW across prediction horizons k. Discovery agents (DA) on capable models outperform prompted baselines at every horizon; on weaker open-weight models, DA performance is bounded by the model's tool-use capability.

Method	**Models**	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
Prompted	GPT-5 Mini	0.190	0.142	0.139	0.094	0.075
	GPT-5.1	0.243	0.190	0.216	0.156	0.149
	Sonnet 4.5	0.358	0.211	0.238	0.199	0.134
	Gemini 3 Pro	0.366	0.238	0.245	0.192	0.130
	Opus 4.6	0.395	0.253	0.228	0.197	0.105
Learned	Qwen-3.5-27B	–	–	–	–	–
	Gemma-4-31B	–	–	–	–	–
Discovery (ours)	Gemma-4-31B	0.214	0.201	0.154	0.134	0.095
	Qwen-3.6-27B	0.227	0.210	0.175	0.189	0.109
	GPT-5.2	0.231	0.187	0.201	0.149	0.091
	GPT-5 Mini	0.258	0.222	0.200	0.164	0.101
	GPT-5	0.291	0.256	0.239	0.191	0.150
	Gemini 2.5 Pro	0.315	0.216	0.239	0.202	0.120
	Gemini 3 Pro	0.369	0.291	0.279	0.221	0.149
	Sonnet 4.5	0.406	0.326	0.329	0.274	0.186
	Sonnet 4.6	0.438	0.314	0.296	0.275	0.189
	Opus 4.6	**0.448**	**0.352**	**0.335**	**0.301**	**0.199**

B Discovery Agent Scores

Table 3 reports the discovery agent's performance on WoW across all evaluated models and prediction horizons $k = 1, \ldots, 5$, alongside the corresponding prompted baselines on the same models. The DA improves over the matched prompted baseline at every horizon on every model where both are evaluated. The improvement is largest at intermediate horizons ($k = 2$ to $k = 4$), where small static-context errors begin to compound but the discovery agent's retrieval still tracks the evolving state.

C Tier-Stratified Results

We stratify CascadeBench results along the three tiers introduced in §3 (Table 5). The tier definitions in §3 are conceptual; here we operationalize them for measurement: T1 covers schema-determined effects on the action's own table, which exercise the data-dictionary defaults and constraints described in Tier 1; T2 covers cross-table cascades requiring at least one business rule to fire, instantiating the rule-composable cascades of Tier 2; T3 covers multi-rule conflicts where two or more rules write distinct values to the same field, an audit-log-detectable instance of the execution-inferred dynamics described in Tier 3. T1 and T2 are scored under IoU(T+F); T3 is scored under Strict IoU on $(\text{table}, \text{field}, \text{value})$, since the defining question for T3 is which conflicting value is realized. System metadata fields, datetime values, and reference identifiers are excluded symmetrically from ground truth and predictions, as they reflect platform internals rather than the business logic under evaluation. Bootstrap 95% confidence intervals ($n = 2{,}000$ resamples) are reported in the supplementary materials.

T1 is predictable from the action and schema alone. Direct IoU on T1 is consistent across all eight models at 0.56–0.60. Schema following is sufficient for action-table effects, and rule retrieval is not required. T2 and T3 drop to 0.00 uniformly under Direct prompting, identifying business rules as the load-bearing signal in CascadeBench.

Discovery exhibits graded degradation across tiers. Under the Discovery Agent, mean IoU decreases monotonically from T1 (0.648) through T2 (0.635) to T3 (0.524). The Oracle condition is

Table 4: Tier-stratified IoU on CascadeBench ($k = 1$, 37 trajectories; T1: 177 keys, T2: 424 keys, T3: 138 keys). *Direct*: prompted without retrieval and without rules. *Discovery Agent*: rules retrieved at inference. *Oracle*: rules in prompt.

Model / Setting	ALL	T1	T2	T3
Direct (no rules, no retrieval)				
Opus 4.6	0.175	0.562	0.043	0.000
Opus 4.7	0.155	0.593	0.000	0.000
Sonnet 4.6	0.152	0.580	0.000	0.000
GPT-5	0.152	0.580	0.000	0.000
Gemini 3 Pro	0.153	0.584	0.000	0.000
Qwen-3.5-27B	0.151	0.578	0.000	0.000
Qwen-3.6-27B	0.151	0.578	0.000	0.000
Gemma-4-31B	0.157	0.601	0.000	0.000
Discovery Agent (rules retrieved at inference)				
Opus 4.6	0.643	0.641	0.635	0.551
Opus 4.7	0.644	0.638	0.635	0.537
Sonnet 4.6	0.638	0.662	0.618	0.482
GPT-5	0.646	0.642	0.639	0.551
Gemini 3 Pro	0.656	0.657	0.650	0.497
Oracle (rules in prompt)				
Opus 4.6	0.635	0.628	0.622	0.578
Opus 4.7	0.627	0.637	0.611	0.523
Sonnet 4.6	0.605	0.581	0.601	0.558
GPT-5	0.691	0.679	0.686	0.599
Gemini 3 Pro	0.659	0.647	0.654	0.588
Qwen-3.5-27B	0.669	0.646	0.669	0.554
Qwen-3.6-27B	0.660	0.635	0.661	0.572
Gemma-4-31B	0.694	0.663	0.695	0.549

flatter (T1: 0.634, T2: 0.635, T3: 0.569), indicating that the T1→T2 gap under DA reflects retrieval overhead rather than reasoning difficulty.

Discovery reaches Oracle parity on T1 and T2; T3 bounds both. Across the five frontier models evaluated under both conditions, mean DA $-$ Oracle deltas are $+0.014$ on T1, $+0.001$ on T2, and -0.046 on T3. The Discovery Agent matches the rule-oracle on the first two tiers without ground-truth rule pre-loading. Both methods plateau on T3, where outcomes depend on execution-order resolution that is not exposed in the configuration either method reads. This is the empirical realization of the boundary anticipated in §3: dynamics determined by execution semantics cannot be recovered from configuration alone.

Open-weight checkpoints reach the Oracle ceiling when rules are supplied in prompt (Qwen-3.5-27B, Qwen-3.6-27B, and Gemma-4-31B all reach ALL IoU ≥ 0.66), indicating the determining factor in this regime is rule content rather than model scale. Whether a discovery loop on these checkpoints can reach the same ceiling depends on tool-use capability, discussed in §A.

D CascadeBench: Construction Pipeline

CascadeBench is generated by a three-stage pipeline that runs against a live ServiceNow instance: schema generation, business rule cascade construction, and cascade execution with audit capture. Each stage combines LLM-driven design with programmatic validation and live execution, and only fully validated, executable examples are included in the benchmark.

Table 5: **Complexity tiers of transition dynamics in enterprise systems.** Transitions range from fully determined by schema (Tier 1), to requiring composition of explicit rules (Tier 2), to behaviors that can only be inferred through execution (Tier 3).

Tier	Description
Tier 1 *Schema-Deterministic*	Transitions fully determined by the data dictionary (field types, defaults, constraints, choice lists); no rule inspection required. **Example:** Creating a user sets: `active=true`, `notification=2`, `locked_out=false`.
Tier 2 *Rule-Composable*	Transitions determined by composing business rules; predictable given full rule knowledge, but requiring tracing of cascading execution. **Example:** Setting priority to P1 triggers auto-assignment, starts an SLA timer, sends a notification, and creates an escalation record.
Tier 3 *Execution-Inferred*	Transitions depend on undocumented engine behavior, timing, or emergent interactions not recoverable from configuration alone. **Example:** Race conditions between synchronous and asynchronous rules produce outcomes only observable through execution.

Schema generation. An LLM proposes a synthetic enterprise domain (e.g., a vendor procurement workflow) and a corresponding schema: up to 10 tables with custom `u_`-prefixed fields, choice values, and foreign-key relationships. Reserved namespace prefixes used by ServiceNow's product modules (`itam`, `cmdb`, `itsm`, `hr`, and others) are excluded so the schemas cannot overlap with platform tables seen during pretraining. The schema and accompanying seed records pass through deterministic structural validation (referential integrity, type conformance, graph connectivity, and naming constraints). Only schemas that satisfy all programmatic checks are deployed to the instance with auditing enabled and consistent seed data.

Business rule cascade construction. Each example contains a cascade of 3–7 business rules that fire from a single triggering action. We support three cascade topologies: *flat* (all rules fire from the same action on the primary table), *linear* (each rule fires on a table written by a prior rule), and *complete graph* (rules may fire on any table any prior rule has written to). Cascades are generated using weighted sampling over rule attributes (trigger type, conditions, fields updated, and tables impacted) to match realistic distributions. For each rule slot, an LLM designs the rule and its script. The script is statically analyzed to extract write operations, which serve as the source of truth for validation. Each rule is validated against 14 deterministic checks covering schema correctness, cascade integrity (including cycle detection), filter validity, and script safety. Failed rules are iteratively repaired; only rules that pass all checks and are verified through execution are included in the cascade.

Cascade execution and audit capture. Once the cascade is constructed, the pipeline deploys the rules to the instance, inserts any supporting records needed by the rule logic, and executes the triggering action. The platform's built-in audit log captures every field-level change that occurs, while a separate custom log traces each change back to the specific rule that caused it. Together, these two logs establish ground truth: one records *what* changed, the other records *why*. Before inclusion in the benchmark, internal metadata fields are filtered out, retaining only semantically meaningful content changes. After each cascade is captured, the instance is reset to its original seed state so that every example starts from identical initial conditions, preventing state leakage across examples.

D.1 What is real and what is synthetic

Below we describe which parts of CascadeBench are taken directly from the platform and which ones we generate.

The dynamics are real. Every transition is computed by the platform's production rule engine on a live instance. The ground truth comes from the platform's audit log, together with a separate log that records which rule caused each field change. No part of the transition function is simulated.

The surface form is synthetic. Table and field names are generated fresh with `u_` prefixes, and the reserved namespaces of the platform's product modules are excluded. We do this on purpose: it lets us attribute performance to reasoning over the provided rules rather than to recall of familiar tables.

The structure follows real deployments. Rule types, trigger conditions, execution phases, cross-table effects, and the range of 3–7 rules per cascade are modelled on configuration templates and documented platform patterns. An LLM instantiates this structure over the fresh schemas, and a rule is kept only if it passes the 14 deterministic checks and runs successfully on the instance.

Actions are state changes, not interaction traces. An action in CascadeBench is a change to a record's fields. The platform responds to that change and to the deployed rules in the same way whether the change comes from the user interface, an API call, or a workflow. We use tool calls only as a way to make the change, so we make no claim about how realistic any particular sequence of interactions is. What needs to be representative is the space of state transitions, and the structure above is designed for that.

Rule attribution enables depth analysis. Because each field change is traced to the rule that caused it, we can measure how prediction quality falls off along a cascade. To our knowledge, no public dataset records enterprise transitions at this level of detail; most enterprise benchmarks score task outcomes rather than the intermediate cascades that produce them.

E Enterprise Gym: World Construction Details

This appendix describes the construction pipeline behind each world $W = (E, T)$ in the Enterprise Gym (4).

Dependency-ordered pipeline. World construction follows a seven-stage pipeline where each stage depends on outputs from earlier stages, mirroring how real enterprise environments are built. Organizational structure (groups, users, roles) is generated first, followed by configuration database topology (services, infrastructure items, dependencies), then process design (per-domain state machines, routing, escalation), then business rules, access policies, SLA definitions, and finally field-level constraints. The ordering guarantees internal consistency: every entity referenced by a business rule has been materialized in an earlier stage. Rules that reference non-existent groups or infrastructure items produce deployment failures, not training data.

Variety engine. Each archetype is seeded with a distinct organizational personality along multiple axes: automation philosophy (heavy vs. light reliance on rules), technical debt posture (clean vs. accumulated legacy automation), and domain completeness (which operational domains are fully built out vs. minimally configured). These dimensions ensure that two worlds in the same industry and company-size category still produce structurally distinguishable transition functions T, preventing the failure mode where a model trained on superficially diverse environments has effectively seen only one underlying transition function.

Conflict injection. Rule conflicts are modeled at three levels: the pattern catalog encodes which rule types interfere with one another, variety profiles set per-archetype conflict density, and the generation stage produces specific conflicting rule pairs with explicit order-dependent behavior. This produces Tier 3 dynamics in the training data, transitions whose outcome depends on platform-internal execution ordering rather than on any single configuration artifact.

State-space augmentation. From approximately 27,000 base scenarios generated by an LLM, a programmatic augmentation step produces $\sim$802,000 total initial states by swapping assignment groups, urgency/impact combinations, caller identities, and infrastructure references drawn from each archetype's validated entity pools. Seven quality guardrails ensure augmented scenarios remain internally consistent: pool validity (every substituted entity exists in the archetype), schema consistency, priority matrix coherence (urgency-impact-priority combinations match the archetype's priority calculator), referential integrity, business rule stripping during bulk insertion (to prevent rule-induced state changes from contaminating initial conditions), deduplication, and a final validation pass.

Table 6: CascadeBench (oracle) vs. WoW (no context) IoU. The gap is the headroom that runtime retrieval can recover.

Model	CascadeBench	WoW	Gap
Sonnet 4.6	38.15	38.65	+0.5
Opus 4.6	40.46	41.32	+0.9
Gemini 3 Pro	41.36	35.59	−5.8
Gemma-4-31B-FT	41.32	31.73	−9.6
GPT-5	41.78	29.34	−12.4
GPT-5.2	38.40	23.97	−14.4
Qwen-3.5-27B	40.39	21.22	−19.2
Qwen-3.5-FT	50.90	31.21	−19.7

F WoW vs CascadeBench

CascadeBench provides every model with the full set of relevant business rules and supporting context, simulating an oracle in which retrieval is perfect. A model's CascadeBench score therefore represents the upper bound on what a discovery agent built around that model could achieve. WoW evaluates the same prediction task on real ServiceNow instances with no provided context, so the gap to CascadeBench measures how much real-world performance is left on the table by imperfect grounding—precisely what discovery agents are designed to recover.

Table 6 shows that this gap is large and model-dependent. Claude Sonnet 4.6 and Opus 4.6 close it almost completely (gaps of 0.5 and 0.9 points), indicating their WoW failures are bounded by reasoning capacity rather than missing context. GPT-5 and Qwen models show 12–19 point gaps, suggesting substantial headroom that runtime retrieval should close. The discovery agent's target is the oracle score; the gap to it is its remaining work.

G Discovery Agent Implementation

We implement the discovery agent as a ReAct-style Yao et al. [2022] agent that predicts enterprise state transitions by inspecting the live system configuration at inference time. Unlike the learned world model, which relies on parameters learned from offline transition data, the discovery agent is given a proposed action and can query the active ServiceNow instance to recover the rules and records needed to predict its effects.

Input preparation. Each prediction begins with an initialization step that loads static context and validates the required state fields. The input state is normalized into JSON-serializable strings to ensure that records, dictionaries, and nested structures are represented consistently across examples. If a step index is not provided, we assign a default value so that multi-step trajectories and single-step examples share the same interface. The resulting prompt contains the table schemas, tool specification, previous record states, and the action whose effects must be predicted.

Agent setup. The discovery agent is composed as a SyGra [Pradhan et al., 2025] graph—an open-source framework that orchestrates LLM workflows—and instantiated as a ReAct agent with a single tool, `snow_query`. The system prompt frames the model as an ITSM domain expert and instructs it to reason about how ServiceNow configuration artifacts determine state transitions. The agent is allowed up to 15 recursive tool calls. This budget is intended to support multi-hop discovery, where predicting a transition may require inspecting business rules, the current record state, choice lists, and SLA definitions before producing a final answer.

Runtime discovery loop. During inference, the agent alternates between reasoning and calls to `snow_query`. The tool provides a uniform interface for querying ServiceNow tables relevant to the prediction task. In practice, the agent most commonly queries four classes of information:

1. **Business rules** from `sys_script`, which specify server-side transition logic triggered by record inserts or updates.

2. **Current record state** from the target task table, which grounds the prediction in the active values of the affected record.
3. **Choice values** from `sys_choice`, which define valid categorical values and help interpret field-level updates.
4. **SLA definitions** from `contract_sla`, which capture additional transition logic related to task timing, priority, and service-level behavior.

The agent uses these queries to identify which rules may fire for the proposed action, determine whether their conditions are satisfied, and infer the resulting field-level updates. The final agent response is required to contain a JSON object with an `audits` field, where each audit entry represents a predicted field-level change.

Output format. The agent predicts transitions in the same field-level diff format used by the benchmark. Each predicted audit record contains the affected table, field name, old value, and new value. This shared output format makes the discovery agent directly comparable to learned world models and prompted baselines. A typical output has the following structure:

```
{
  "audits": [
    {
      "tablename": "...",
      "fieldname": "...",
      "oldvalue": "...",
      "newvalue": "..."
    }
  ]
}
```

Post-processing. Because the final response is produced by a language model, we apply a deterministic post-processing step before scoring. The `DiscoveryPostProcessor` first attempts to extract JSON from fenced code blocks. If no fenced JSON block is found, it falls back to a greedy `{.*}` regular expression, matching the extraction behavior used in WoW. Each predicted audit entry is then normalized to the expected schema, retaining the canonical fields `fieldname`, `newvalue`, `tablename`, and `oldvalue`. Finally, the normalized prediction is written to the task state as `predicted_state_json`.

Design motivation. This implementation intentionally exposes only a minimal query interface rather than a large collection of hand-engineered tools. The goal is to test whether an agent can recover transition logic from the same configuration artifacts that define the live enterprise system. By grounding its prediction in the current deployment, the discovery agent can adapt to tenant-specific rules and configuration changes without requiring retraining.

Pseudocode. The algorithm below consolidates the components described above into a single procedure: the outer autoregressive rollout, the inner ReAct loop with the `snow_query` tool, and the deterministic post-processor.

Algorithm 1: Discovery Agent — autoregressive state-prediction rollout

```
Input:  trajectory ((a_1, s_1), ..., (a_K, s_K)); compressed schemas
        Sigma, tool specs Tau; live instance configuration c.
Output: predicted per-step diffs hat_s[1:K], where each hat_s[t] is the
        set { (tablename, fieldname, oldvalue, newvalue) }.

1. Outer rollout (autoregressive over trajectory steps)
hat_s := []
for t := 1, ..., K do
    state := InitState(a_t, hat_s, Sigma, Tau, step_index = t)
    msg   := DiscoveryAgent(state)
    hat_s.append( Postprocess(msg) )
```

```
return hat_s

2. DiscoveryAgent: ReAct loop, single tool
DiscoveryAgent(state):
    ctx := SystemPrompt + Render(state)
    while not FinalAnswer(msg) do
        msg := f_LLM(ctx)
        for (table, query, fields, limit, order_by) in msg.tool_calls do
            result := snow_query(table, query, fields, limit, order_by)
            ctx    := ctx + (msg, result)
    return msg

3. Postprocess: extract {audits: [...]} from the final message
Postprocess(msg):
    obj := ExtractFencedJSON(msg.content)
    if obj is None then
        obj := GreedyJSONMatch(msg.content)
    if obj is None or 'audits' not in obj then
        return { audits: [] }
    return { audits: [
        {tablename, fieldname, oldvalue, newvalue} : a in obj.audits ] }
```

H CascadeBench: Failure Mode Analysis

To characterize the gap between the oracle ceiling (prompted models with full business rules in context, 38–50% IoU on CascadeBench) and perfect cascade prediction, we manually analyze two representative trajectories. Both are evaluated under the oracle condition: the model receives the schema, supporting data, and all relevant business rules in context, and is asked to predict the field-level cascade. We identify three recurring failure modes that account for the bulk of missed predictions even when context is complete.

Failure patterns:

P1 | Insert/creation blindness. When a business rule calls `gr.insert()`, the new record's fields produce 7–12 auditable changes. The model omits the insert entirely or predicts only 1–3 salient fields, missing the rest.

P2 | Cascade fade-out. The model traces BR1–BR2 at 75–85% recall but drops to 4–11% for business rules at execution order ≥ 400. Entire tables produced by deep-cascade rules are absent from predictions.

P3 | Single-record assumption. When a business rule iterates a result set (`while (gr.next())`) or issues multiple inserts, the model predicts effects on exactly one record per table, missing parallel updates and sibling inserts.

These are reasoning patterns, not retrieval patterns: the rules and supporting context are present in the prompt. They characterize a ceiling on what any context-fed approach (prompted oracle or perfect-retrieval discovery agent) can achieve without explicitly training the model to compose multi-step rule cascades.

Trajectory 1: Accounts Payable — Invoice Match Status

Action. `UPDATE` on `u_ap_vendor_invoice` setting `u_match_status = 3` (Fully Matched).

Cascade structure. A 6-rule cascade spans vendor invoices, purchase orders, line items, approval requests, and payment disbursements.

```
UPDATE vendor_invoice.match_status -> 3
|
+-- BR1 [order 100] vendor_invoice, purchase_order      [predicted well]
```

```
+-- BR2 [order 200] purchase_order, INSERT approval     [predicted well]
+-- BR3 [order 200] vendor, po_line_items(s)            [partial - P3]
+-- BR4 [order 400] po_line_items(s)                    [partial - P3]
+-- BR5 [order 500]
|   |-- invoice_line_item(s)                           [missed - P1+P2+P3]
|   |-- INSERT payment_disbursement                    [missed - P1]
|   |-- approval_request(s)                            [missed - P2+P3]
+-- BR6 [order 600]
    |-- invoice_line_item                              [missed - P2]
```

Result. 22 predictions, 15 correct (precision 68%, recall 34% against 44 ground-truth audits). The model traced BRs 1–2 accurately but missed the `u_ap_invoice_line_item` table entirely (17 missed changes from BRs 5–6). Multi-pass overwrites on `u_gl_account_code` were captured only for the first pass.

Trajectory 2: Contract Obligation — Overdue Escalation

Action. `UPDATE` on `u_cot_obligation` setting `u_status = 5` (Overdue) and `u_priority = 4` (Critical).

Cascade structure. A 5-rule cascade spans obligations, contracts, counterparties, escalations, audit logs, and reviews.

```
UPDATE obligation (status -> 5, priority -> 4)
|
+-- BR1 [order 100]                                    [predicted well]
|   INSERT escalation (7 fields)
|   INSERT audit_log #1 (11 fields)
|   UPDATE contract (3 fields)
+-- BR2 [order 200] contract overwrite                 [predicted well]
+-- BR3 [order 200]                                    [missed - P2]
|   counterparty fields, INSERT observer counterparty
+-- BR4 [order 400] observer defaults                  [missed - P2]
+-- BR5 [order 500]                                    [missed - P2+P3]
    UPDATE escalations (nested while)
    INSERT audit_log #2
    UPDATE reviews (while)
```

Result. 26 predictions, 14 correct (precision 54%, recall 29% against 48 ground-truth audits). The model traced BRs 1–2 accurately (86–100% of their outputs) but predicted 0% of outputs from BRs 3–5. It produced one audit log record where two exist, and entirely missed `u_cot_obligation_review` (9 changes) and `u_cot_counterparty` (2 changes).

H.1 Aggregate Failure Analysis

Table 7: Recall by cascade depth across both trajectories. Early business rules (depth 1–2) are well covered; deep rules (depth ≥ 3) are nearly entirely missed.

BR depth	AP Invoice	COT Obligation
Depth 1–2 (order 100–200)	12/16 (75%)	17/20 (85%)
Depth ≥ 3 (order ≥ 400)	3/28 (11%)	1/28 (4%)

P1: Under-prediction of record creation. A single missed insert produces 7–12 false negatives at once. Creation-phase audits (`old_value = ""`) are recalled at 24–27%, roughly half the rate of update audits (36–47%), confirming the model is systematically weaker at predicting record creation than record modification.

P2: Cascade coverage drops after the first two rules. Table 7 shows the model simulates rule scripts accurately for the first 1–2 hops but does not sustain this across longer chains. Entire tables

Table 8: False-negative attribution by failure pattern. Many false negatives carry multiple tags.

	Description	AP	COT
P1	Insert/creation-phase audits missed	13	15
P2	Cascade fade-out (depth > 2)	19	20
P3	Single-record assumption	5	8
Other	Boolean coercion, ref-key, timestamp, value	5	5

disappear from predictions, and multi-pass overwrites are half-predicted: when two rules write the same field sequentially, the model captures only the first write.

P3: Single-record assumption. The model treats each table as having a single "affected record." AP BRs 3–5 use `while (gr.next())` to iterate all PO line items, invoice line items, and approval requests; the model predicted changes to at most one record per table. COT BR3 updates the existing contract counterparty and inserts a new observer counterparty record; the model predicted only the update, not the sibling insert.

Implications. These failure modes are reasoning bottlenecks of the underlying model, not retrieval failures: they persist when all rules are provided in the prompt. Closing the gap to perfect cascade prediction therefore requires training the model to compose rule executions, not just to retrieve them. This is the direction of the trained discovery agent proposed in §6.

I Fine-tuning details

We fine-tune three recent open-weight language models with strong agentic benchmark performance: Qwen-3.5-27B [Qwen Team, 2026a], Qwen-3.6-27B [Qwen Team, 2026b], and Gemma-4-31B-it [Google DeepMind, 2025]. We apply LoRA [Hu et al., 2022] with rank 16 and $\alpha = 32$ to the state-transition tuples from §4.

All models are trained for 2 epochs with a global batch size of 32 using AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.01) and a cosine learning rate schedule with 10% linear warmup; learning rates are selected via a held-out validation set: 1×10^{-4} for Qwen and 2×10^{-4} for Gemma-4-31B-it. During fine-tuning, the maximum sequence length is set to 32k tokens for Qwen models and 12k tokens for Gemma-4-31B-it.

J Glossary

The following terms describe enterprise platform concepts referenced throughout the paper.

Business Rule
(ServiceNow: *Business Rules or BRs*) A server-side script that executes automatically when a record is created, updated, or deleted. Each rule has a trigger condition, an execution phase, and a numeric priority that determines firing order relative to other rules on the same table.

Cascade
A chain reaction in which one business rule's output triggers another rule, which may trigger further rules. A single user action can produce cascades spanning multiple tables and dozens of intermediate rule firings.

Configuration database
(ServiceNow: *CMDB — Configuration Management Database*) A structured repository of managed assets (servers, applications, network devices, software services) and their dependency relationships. Changes to one asset can propagate effects to all dependent assets.

Execution phase
(ServiceNow: *Business Rule "When" field*) The stage at which a business rule fires relative to the database operation: *before* (can modify the record before it is written), *after* (runs once the write is committed), *async* (runs in a background thread), or *display* (runs when the record is loaded for viewing).

Instance
A single deployed installation of the enterprise platform, configured with its own business rules, access policies, and organizational structure. Different customers operate different instances with different configurations.

Instance configuration (c)
The full collection of business rules, workflow definitions, approval policies, SLA definitions, and access control policies deployed on a particular instance. This is what makes the transition function instance-specific.

Service-level agreement (SLA)
(ServiceNow: *contract_sla* and *task_sla*) A policy defining response and resolution time targets for tasks. When an SLA's start condition is met, the platform creates a timer record that tracks elapsed time and can schedule escalations.

Access control policy
(ServiceNow: *ACL — Access Control List*) A rule governing which users or roles can read, write, or delete records on specific tables or fields. These can silently block or modify the effect of agent actions.

Audit log
(ServiceNow: *sys_audit*) A platform-maintained record of every field-level change, including the old value, new value, timestamp, and the identity of the actor. This is the ground-truth source for capturing state transitions (s_t, a_t, s_{t+1}).

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's claims: offline-trained world models perform well in distribution but degrade under configuration shift; discovery agents that read the active configuration at inference time are more robust; and rule access is necessary but not sufficient, with multi-step rule composition rather than retrieval as the binding constraint. These claims are supported in Section 6 (Tables 1 and 2) and Appendix H. The scope (a single platform with inspectable configuration) is stated in Section 9.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: A dedicated Limitations section (Appendix A) discusses inspectability assumptions, tool-use capability bounds on discovery agent performance, single-platform evaluation scope, and the small-sample nature of the same-backbone DA-vs-trained comparison.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper presents an empirical study and does not include theorems or formal proofs. The contextual transition model in Section 3 is a problem formulation rather than a theoretical result.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 5 describes the prompted baseline, the LoRA-finetuned world model (training data source, target format, fine-tuning configuration), and the discovery agent architecture. Section 4 and Appendix E describe the Enterprise Gym corpus construction. Section 6 reports all evaluation conditions, metric definitions, and per-model results. Appendix H provides per-trajectory diagnostic detail.

   Guidelines:

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

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

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

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

   Answer: [Yes]

   Justification: CascadeBench is publicly available at `https://huggingface.co/datasets/ServiceNow-AI/cascade_bench` under the Apache-2.0 license, and the discovery agent is released in SyGra (`https://github.com/ServiceNow/SyGra/tree/scratch/ewm/tasks/examples/wow_state_predictor_da`). The dataset construction pipeline and the evaluation harness are not yet public; the construction pipeline is described in Appendix D and the metrics in Section 6. The WoW evaluation requires access to a ServiceNow instance [Gupta et al., 2026].

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 6 specifies the evaluation settings (w/ BR vs. w/o BR on CascadeBench; deployment-shift WoW), the metrics (strict IoU and IoU(T+F)), and per-model results. The fine-tuning setup (LoRA, base models, target format) and the discovery agent's tool surface and prompt template are described in Section 5. Train/test splits are described in Section 4.

   Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: We report point estimates rather than error bars. Each evaluation row aggregates over a substantial number of test instances (CascadeBench: hundreds of examples per condition; WoW: per-trajectory across the held-out split), and the relative ordering between methods is consistent across both benchmarks. Multi-seed runs were not feasible within the inference-cost budget for frontier API evaluations and are deferred to future work.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: The paper does not currently report compute resource breakdowns. Frontier-model evaluations were run via standard hosted API endpoints; LoRA fine-tuning on Qwen-3.5/3.6 (27B) and Gemma-4 (31B) used a small number of A100/H100 GPUs over short training runs. We will include a compute disclosure in the camera-ready version.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

Justification: The work uses synthetic schemas and synthetic enterprise environments throughout. No human subjects, scraped personal data, or real customer data were involved. The research conforms to the NeurIPS Code of Ethics.

Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: Discovery agents that retrieve enterprise configurations rather than internalize them improve auditability and reduce the risk of agents acting on stale or incorrect dynamics, both of which are positive for safe enterprise deployment. Potential negative impacts include increased agent autonomy in enterprise systems where errors can propagate through downstream business processes; we view honest characterization of remaining failure modes (Appendix H) as a step toward responsible deployment.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release pre-trained generative models, image generators, or scraped datasets that pose high misuse risk. The synthetic CascadeBench artifacts and the LoRA adapters described in this paper do not present such risks.

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

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Existing assets used in the paper are cited at point of use: WoW [Gupta et al., 2026] for evaluation methodology, base models from their respective providers (Anthropic, Google, OpenAI, Qwen, Gemma) cited via official model cards or technical reports, and LoRA [Hu et al., 2022] for fine-tuning. All experiments use models accessed through their published APIs or weight releases under their respective terms of use.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: CascadeBench is released with a dataset card on Hugging Face under the Apache-2.0 license; its construction and validation are documented in Appendix D. The Enterprise Gym training corpus and the fine-tuned LoRA adapters are not released; their construction is documented in Appendices E and I.

    Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve research with human subjects, so IRB approval is not applicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: LLMs are central to the methodology. Frontier and open-weight LLMs are the agents under evaluation across all conditions (prompted baseline, learned world model, discovery agent). The Enterprise Gym training corpus is generated using a teacher-model pipeline that combines LLM-generated scenario specifications with deterministic platform-side execution and validation (Section 4, Appendix E). The discovery agent is itself an LLM-driven retrieval-and-prediction loop (Section 5).

Guidelines:

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