EVALUATING OPEN-ENDED DECISION-MAKING WITH LLM-AS-ENVIRONMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We study how to evaluate open-ended decision-making. Because outcomes may take years to become clear and alternative choices cannot be observed, evaluating these decisions requires simulating how different choices shape future outcomes. We introduce **LLM-AS-ENVIRONMENT**, a framework for constructing environments for this purpose. Our key idea is to use an LLM as part of the environment to simulate changes that are difficult to specify with fixed rules, while code handles calculations, constraints, and state updates. We also develop design principles and skill files that guide coding agents in constructing, calibrating, and stress-testing these environments. Using this framework, we build **FUZZYDECISIONBENCH**, with environments for undergraduate career planning, startup strategy, and NFL head coaching. We validate the environments using human judgments, historical data, and a controlled comparison with an existing rule-based simulator. Replacing the rule-based transitions in YC-Bench, a programmatic startup simulator, with LLM-predicted transitions yields a model-ranking correlation of $\rho = 0.857$ with a coded simulator run, comparable to the correlation between two independent coded runs ($\rho = 0.738$). We then evaluate 11 frontier language models across all three environments. Even the strongest models close only 25.1–55.6% of the utility gap between a scripted baseline and a privileged search reference, leaving substantial room for improvement. Advances in LLM capabilities enable a new approach to building decision-making environments: LLMs can help construct them and also serve as part of the environment itself. We release the environments, evaluation harness, and builder so that researchers can refine these environments and construct new ones for other decision problems.

1 INTRODUCTION

Language models are increasingly used to help people make important decisions, yet we lack ways to evaluate how well they do so (Luettgau et al., 2025). Consider a college freshman deciding whether to focus on classes, join a research lab, pursue extracurriculars, or build a startup. In the real world, outcomes of decisions may take years to become clear, and we cannot observe what would have happened under different decisions. Evaluating decisions therefore requires simulating how they shape future outcomes. Existing benchmarks evaluate long-horizon decision-making by simulating environment transitions with hard-coded rules (Backlund & Petersson, 2025; Hu et al., 2026; He et al., 2026). However, real-world decisions often require qualitative reasoning about potential outcomes that cannot be modeled with fixed rules. For example, whether joining a research lab is worthwhile may depend on how well the student's interests match the project, how much guidance the advisor provides, and how the collaboration develops over time.

We call these settings *fuzzy decision-making*: the action space is broad, outcomes are delayed, and action effects resist simple fixed rules. Building evaluation environments therefore requires choosing which variables of the setting to represent and modeling how decisions change them.

We introduce LLM-AS-ENVIRONMENT, a framework for constructing environments for fuzzy decision-making. While existing benchmarks use code-based rules to simulate how the world changes after each action, our key idea is to use an LLM to simulate parts of these changes. To help researchers construct these environments, we develop design principles and encode them in skill files that guide off-the-shelf coding agents through building, refining, and testing an environ-

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

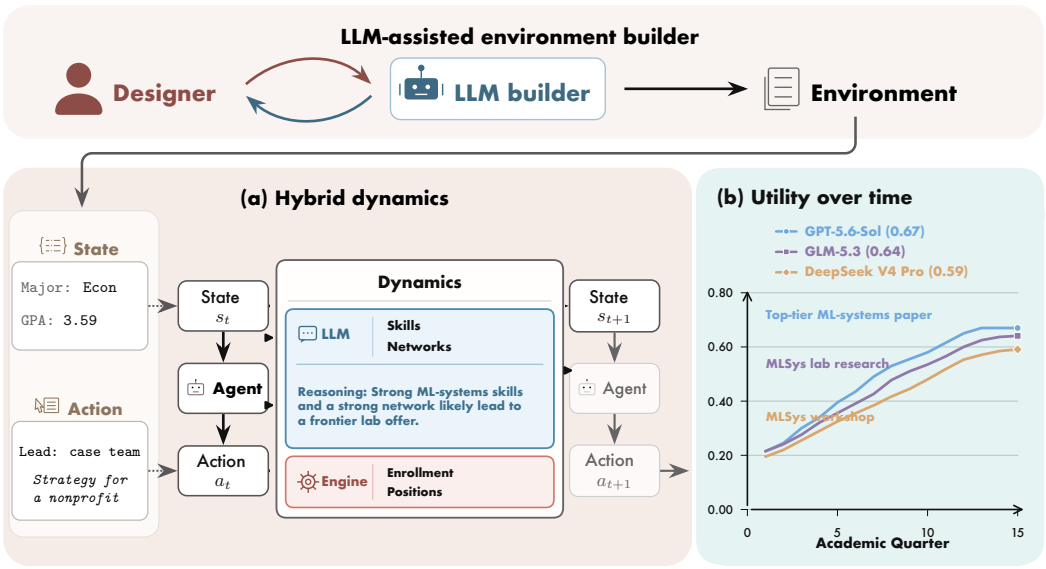

Figure 1: **LLM-AS-ENVIRONMENT supports the construction and evaluation of long-horizon decision environments.** A designer works with an LLM-assisted builder to specify the environment. **(a) Hybrid dynamics:** an LLM reasons about contextual dependencies—illustrated by how ML-systems skills and professional networks affect a frontier-lab offer—while a programmatic engine enforces exact constraints and state updates. **(b) Long-horizon evaluation:** three LLM agents start from the same initial state and produce distinct utility trajectories over 15 academic quarters, with GPT-5.6-Sol achieving the highest final utility.

ment. Using this framework, we construct **FUZZYDECISIONBENCH**, a suite of three environments: College Career Planning, for undergraduate career planning; Startup Strategy, for startup strategy; and NFL Team Management, for NFL head coaching (Figure 1).

We evaluate eleven frontier models, normalizing utility between a scripted baseline and privileged simulator-aware search; the strongest models close 55.6%, 25.1%, and 26.5% of this gap in College Career Planning, Startup Strategy, and NFL Team Management, respectively. We release the environments, evaluation harness, and builder so researchers can refine them and construct new decision environments.

We then validate the environments in FUZZYDECISIONBENCH by collecting human plausibility judgments in College Career Planning and replaying historical decisions in Startup Strategy and NFL Team Management. Furthermore, to stress-test the LLM-AS-ENVIRONMENT framework, we replace YC-Bench's coded dynamics with LLM dynamics (He et al., 2026). Across eight models, rankings under LLM dynamics correlate with a coded YC-Bench run at $\rho = 0.857$, comparable to the correlation between two independent coded runs ($\rho = 0.738$).

We view FUZZYDECISIONBENCH as a step toward better evaluation of fuzzy decision-making with LLM-AS-ENVIRONMENT. Recent advances in LLMs make this possible: LLMs can both help construct these environments and serve as part of the environment itself. We anticipate that the fidelity of these environments will improve as LLM capabilities improve. In summary, we make the following contributions:

- **Framework and construction:** We introduce LLM-AS-ENVIRONMENT, which combines LLM transition predictions with a programmatic engine, and provide design principles and skill files for constructing and stress-testing such environments.
- **FUZZYDECISIONBENCH:** We build environments for undergraduate career planning, startup strategy, and NFL head coaching, and evaluate frontier language models across all three.

- **Environment validation:** We validate the environments through comparison with hard-coded simulators, human evaluations, and historical data.

2 PRELIMINARIES

Our goal is to evaluate complex, real-world decision-making. To make this setting precise, we adopt the standard formalism of partially observable Markov decision processes (POMDPs).

POMDP formulation. Let $\mathcal{S}$, $\mathcal{A}$, $\mathcal{O}$, ρ_0, T, g, U, and H denote the state space, action space, observation space, initial-state distribution, transition function, observation function, utility function, and horizon, respectively. We write $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \rho_0, T, g, U, H)$. At each step t, the agent observes $o_t = g(s_t)$, chooses an action $a_t \in \mathcal{A}(s_t)$, and the environment transitions according to $s_{t+1} \sim T(\cdot \mid s_t, a_t)$. After H steps, the trajectory τ receives utility $U(\tau)$.

An illustrative example. Consider a college freshman pursuing a technology career while maintaining well-being, with one decision per academic quarter. The action space must cover both time allocation and open-ended opportunities such as independent projects or leave, while the state must capture preparation, workload, finances, interests, and commitments without modeling the entire world.

Even then, specifying T is difficult. For example, the effect of taking leave to pursue a startup depends on co-founders, demand, finances, and academics, relationships that admit useful heuristics but no reliable fixed transition rule.

Defining fuzzy decision-making. We refer to decision problems with these two challenges as *fuzzy decision-making*: the relevant state and action spaces do not have an obvious representation, and the effects of actions depend on relationships that are difficult to specify precisely with fixed rules or formulas.

3 LLM-AS-ENVIRONMENT

We introduce LLM-AS-ENVIRONMENT, a framework for constructing environments for fuzzy decision-making that addresses the two challenges above by leveraging LLMs in two complementary ways. First, to help define state and action spaces that capture the relevant aspects of a decision problem, we provide design principles and implementation templates that guide an LLM coding agent as it helps a human designer build the environment. Second, to model transitions whose effects cannot be specified reliably with fixed rules or formulas, we use an LLM within the environment dynamics. Figure 1 summarizes the framework. Throughout this section, we use the college career planning problem from Section 2 as a running example.

3.1 DEFINING THE DECISION PROBLEM

The designer specifies ρ_0, g, U, and H, then uses the LLM environment builder to construct the state and action spaces; Section 4 summarizes these components for each environment.

3.1.1 STATE REPRESENTATION

We match the state representation to the decision cadence, include variables that can affect decisions while omitting derivable quantities, and use types suited to what each variable measures. Where possible, variables use natural units such as GPA; concepts without natural units use documented abstract scores, such as `social_connectedness`=0.7. The resulting `state.yaml` records each variable's type, description, range, initialization, and update mechanism.

3.1.2 ACTION REPRESENTATION

Action granularity likewise matches the decision cadence. Because a fixed menu cannot cover every plausible choice and unconstrained text can let an agent assert desired outcomes, we use general action types with structured arguments and limited free-text fields. For example, `launch_startup`

Table 1: **FUZZYDECISIONBENCH environments.** Counts come from each environment's `state.yaml` and `action.yaml`; full specifications are in the appendix.

	College Career Planning	NFL Team Management	Startup Strategy
Role; horizon	Student's advisor; 15 quarters	Head coach; up to 10 seasons	Founder and CEO; 36 months
State; actions	78 variables; 23 time allocations and 17 structural actions	36 variables; 13 typed strategy fields	25 structured variables; operating plan and 19 decision types
Dynamics LLM decides	Changes in skills, profiles, relationships, and health; outcome and tier of each application	Position-group changes, owner trust, wins, playoff depth, title probability, firing	Execution, customer response, and investor response, in three sequential judgments
Engine handles	Time budget, recruiting windows, offer deadlines, placement limits, graduation	Salary cap, contracts, aging, offseason acquisition draws, Super Bowl draw	Revenue, payroll, cash, debt, cap table, closing of transactions
Utility; evaluation	Career outcome and well-being; 7 objective–state pairs	Expected titles; 5 states $\times$ 3 seeds	Terminal company value; 5 states

records the sector, number of co-founders, pitch, and problem, but the environment—not the agent's wording—determines funding. The corresponding `action.yaml` specifies arguments, allowed values, availability conditions, and resource constraints.

3.2 HYBRID LLM-CODE TRANSITION DYNAMICS

To simulate state transition dynamics, we use a combination of an LLM and a programmatic engine. We use code for state updates governed by explicit rules (e.g., the student's enrollment status) and an LLM for effects that require qualitative judgment (e.g., how a quarter of research affects technical skills). For each state variable, `state.yaml` specifies whether code or the LLM updates it.

A transition has three stages. First, code checks that the action is valid and applies any rule-based effects the LLM should see, such as starting a position the student accepted. Second, the dynamics LLM receives the resulting state, the action, and the definitions of the relevant variables. Its prompt instructs it to treat the action as an attempt rather than an outcome and to account for limited time, neglected activities, and diminishing returns, and it returns only changes to the variables it owns, in a typed schema. Third, code validates these changes, discards changes to variables the LLM does not own, clamps values to their ranges, and applies their rule-based consequences. For example, if the LLM decides that an application succeeds, code creates the corresponding offer. We summarize the transition as

$$\tilde{s}_t = \mathrm{Code}_{\mathrm{pre}}(s_t, a_t), \qquad \Delta_t = \mathrm{LLM}(\tilde{s}_t, a_t), \qquad s_{t+1} = \mathrm{Code}_{\mathrm{post}}(\tilde{s}_t, \Delta_t, a_t).$$

4 FUZZYDECISIONBENCH

We instantiate LLM-AS-ENVIRONMENT in three decision problems: College Career Planning, NFL Team Management, and Startup Strategy. The environments differ in real-world context, time horizon, constraints, and objectives, but in each one the agent makes a sequence of decisions whose consequences depend on the evolving state. Table 1 summarizes the three environments. Section 4.1 describes how we built and refined the environments, and Sections B.1, C.1 and D.1 describe each one in detail.

4.1 BUILDING THE ENVIRONMENTS

Drafting. Following Figure 1, a designer describes the decision problem in natural language, including who makes the decisions, what counts as a good outcome, the decision cadence and hori-

Table 2: **Main results.** Mean normalized utility $\widetilde{U}$ for the 11 models evaluated in all three environments. Raw and state-specific results are in the appendix.

Agent	College Career Planning	Startup Strategy	NFL Team Management
GPT-6 Astra	**55.6%**	21.0%	**26.5%**
Claude Opus 5.5	51.0%	19.3%	17.8%
GPT-5.6 Sol	50.6%	15.7%	20.6%
Gemini 3.8 Flash	48.8%	8.5%	19.4%
Kimi K3	48.8%	8.5%	21.4%
Claude Fable 5.1	47.8%	**25.1%**	17.9%
GLM-5.3	44.8%	3.0%	19.1%
DeepSeek V4 Pro	39.8%	7.9%	15.8%
GPT-5.6 Luna	39.3%	7.3%	17.5%
Qwen3.8-Max	39.3%	0.3%	17.8%
Claude Sonnet 5	28.0%	3.2%	15.1%

zon, and the main tradeoffs. A coding agent guided by our skill files then drafts the state and action schemas, utility function, initial-state sampler, dynamics prompt, engine, and test scenarios.

Testing and freezing. We refine each environment using these scenarios and the exploits exposed by the privileged searches that produce our upper reference policies (Sections B.6 and 5.1). Before evaluation, we freeze the dynamics prompt, engine, and initial states so that every agent and both reference policies use the same environment (Section A.2). Because development scenarios shape the environment, we exclude them from the independent validation evidence in Section 6.

5 RESULTS

We next evaluate frontier LLMs in the three environments in FUZZYDECISIONBENCH. In addition to utility, we analyze the decisions within each trajectory to identify where models succeed and fail.

5.1 EXPERIMENTAL SETUP

Models and agent interface. We evaluate GPT-6 Astra, GPT-5.6 Sol, GPT-5.6 Luna, Claude Opus 5.5, Claude Fable 5.1, Claude Sonnet 5, Gemini 3.8 Flash, Kimi K3, GLM-5.3, DeepSeek V4 Pro, and Qwen3.8-Max. At each step, the agent observes U, $g(s_t)$, and available actions, but not the implementation, dynamics prompt, engine code, or hidden bookkeeping variables, and selects a_t.

Evaluation protocol. Within each environment, all policy models use the same dynamics LLM. We use GPT-5.5 with medium reasoning and temperature 0 for seven College Career Planning objective–state pairs and five NFL Team Management states over three seeds, and GPT-5.6 Sol with medium reasoning and temperature 0 for five Startup Strategy companies.

Utility normalization and reference policies. Because raw utility has no natural lower or upper bound, we compare each agent's policy π against two reference policies. The lower reference, π_{lower}, is a simple, hand-coded policy. The upper reference, π_{upper}, is the highest-utility trajectory found by a privileged coding-agent search that can inspect the environment implementation and dynamics prompt, observe simulator-private state and outcomes, and execute and compare multiple candidate trajectories. In NFL Team Management, the privileged search can also inspect the seeded random draws. Let $U_{i,r}(\pi)$ denote the utility obtained by policy π from initial state i under seed r. For initial state i and seed r, we define normalized utility as

$$\widetilde{U}_{i,r}(\pi) = \frac{U_{i,r}(\pi) - U_{i,r}(\pi_{\text{lower}})}{U_{i,r}(\pi_{\text{upper}}) - U_{i,r}(\pi_{\text{lower}})}.$$

We report normalized utility $\widetilde{U}$ averaged over initial states and seeds.

5.2 MAIN RESULTS

Table 2 reports the aggregate results. We summarize our main findings.

Frontier models leave substantial headroom for better decision-making. GPT-6 Astra ranks first in College Career Planning and closes only 55.6% of the gap between the scripted reference and privileged search, followed by Claude Opus 5.5 and GPT-5.6 Sol. GPT-6 Astra also ranks first in NFL Team Management, where it closes 26.5% of the reference gap, followed by Kimi K3 and GPT-5.6 Sol. In Startup Strategy, Claude Fable 5.1 has the highest mean normalized utility and closes 25.1% of the reference gap, followed by GPT-6 Astra at 21.0% and Claude Opus 5.5 at 19.3%; all other models close less than 16%.

Performance depends on planning horizon. In a matched College Career Planning ablation, GPT-6 Astra with the full 15-quarter horizon reaches 55.3% normalized utility, compared with 14.2% for a policy that optimizes only the next transition. The full-horizon policy wins all seven objective–state pairs (two-sided exact sign test, $p = 0.0156$). Adding an explicit persistent plan does not improve the result (55.1%; Section B.5).

5.3 Qualitative Analysis

Aggregate scores do not show which decisions produce the final utility. We therefore compare model trajectories with the scripted and privileged reference policies in one revealing initial state from each environment (Figure 2). Reviewers can inspect the available trajectories—including per-step observations, agent decisions, state transitions, and dynamics reasoning—in an anonymous interactive viewer.

College Career Planning: building evidence before applying. On the technology-career objective, the scripted policy raises the profile score from 0.160 to 0.312 through standard coursework but finishes with zero interview readiness and zero career outcome. GPT-6 Astra and Claude Opus 5.5 obtain utilities of 0.687 and 0.684 by following broad software-engineering paths: technical coursework, internships, and applications to backend, systems, or ML-systems roles. Privileged search instead links an ML-systems project to research, a paper, an ML-infrastructure internship, and a related application, reaching a profile score of 0.672, readiness of 0.865, and an outcome of 0.90. Its advantage comes from producing evidence for specific later opportunities, not simply from increasing preparation.

NFL Team Management: surviving is necessary but not sufficient. The young-and-cheap initial state has substantial salary-cap space but no elite position group. The scripted reference makes no optional roster moves and is fired after three consecutive 8–9 seasons without a playoff appearance. GPT-6 Astra and Claude Fable 5.1 avoid firing and reach a Super Bowl, but spread acquisitions across several position groups and finish with only 0.38–0.40 expected titles. Privileged search concentrates early signings, trades, and draft selections at linebacker while continuing to draft quarterbacks; it reaches seven Super Bowls, wins five, and finishes with 4.17 expected titles. This result is simulator-specific: privileged search considers 186 candidates and knows the random draws, so it does not show that concentrating on linebackers works in general.

Startup Strategy: adapting the product, but exploiting the metric. The scripted reference follows a generic workflow and ends with one financing, \$0.24 million in annualized revenue, and \$25.8 million in utility. Claude Opus 5.5 chooses a managed RL post-training service and reaches \$6.77 million in revenue but makes only one material product update after launch. Privileged search instead builds a code-repair system with executable tests and later adds task decomposition and budget-aware stopping when multi-step maintenance proves too expensive. Yet its utility advantage comes mainly from nine financings and a \$1.6 billion valuation despite only \$3.54 million in revenue, compared with Claude Opus 5.5's \$510 million valuation. This exposes a benchmark failure: utility uses the latest priced-round valuation without charging for founder dilution, so repeated fundraising can dominate realized revenue.

Common failure modes. The three examples show recurring problems. In College Career Planning, similar profile scores can produce different outcomes when preparation does not support later applications. In NFL Team Management, 16 of 39 young-and-cheap trajectories (41%) and 7 of 39 rebuild trajectories (18%) end in firing, while none do in the other three scored states. In Startup

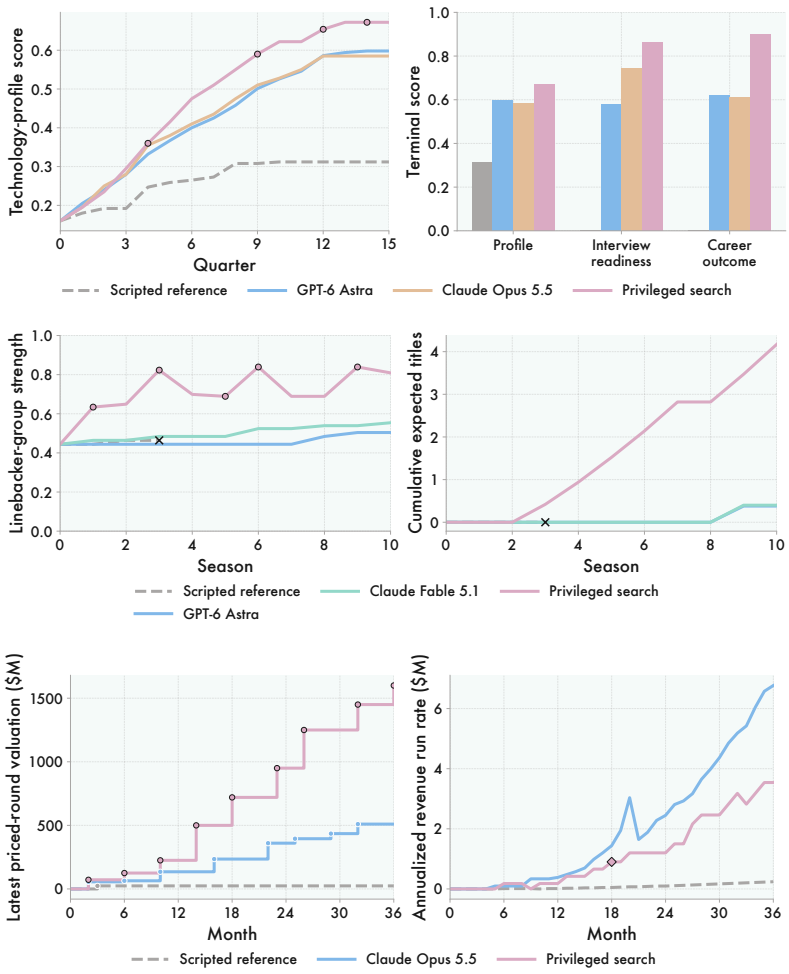

Figure 2: **Representative trajectories in the three environments.** Top: career preparation and outcomes; middle: NFL performance and roster investment; bottom: startup revenue and valuation. Markers identify the decisions discussed in the text.

Strategy, similar revenue can yield very different priced-round valuations, and repeated financing can raise utility without comparable operating progress.

6 Validating the Environments

Because these decision problems lack ground-truth simulators, we use four tests: replacing known dynamics with an LLM, judging transition plausibility, replaying historical decisions, and testing sensitivity to the dynamics LLM.

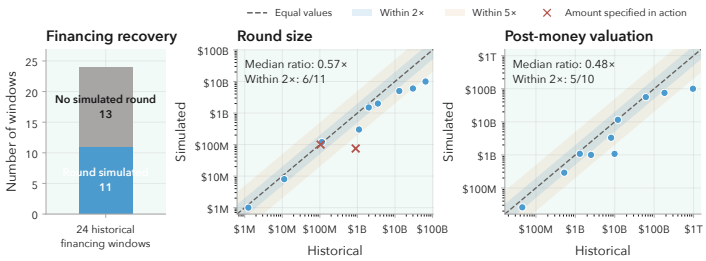

Figure 3: **Startup financing replay.** A separate replay with GPT-5.6 Sol dynamics recovers 11 of 24 financing windows. For matched rounds, median simulated-to-historical ratios are 0.57 for amount and 0.48 for post-money valuation; bands mark $2\times$ and $5\times$ agreement.

Table 3: **Environment validation results.** ρ is Spearman rank correlation. We discuss experiment details in Sections B.6, C.4, D.4, D.5 and E.3.

Test	Env.	Result	Comparison
Fixed-action and agent-ranking replay	YC-Bench	3.91% funds error; 23/24 directions; rank $\rho = 0.857$	Two coded runs: $\rho = 0.738$
Transition plausibility and agreement	Career	Judges: 89–94% direction, 86–92% magnitude	Human–human: 95.4% direction, 91.4% magnitude, 44.4% coherence
Historical action replay	Startup	99/106 financing decisions; 13/18 statuses	Median error: $2.0\times$ amount, $2.36\times$ post-money
Historical replay: wins and titles	NFL	Win MAE 3.30; title $\rho = 0.825$	Betting-line MAE 2.27
Swap dynamics LLM	NFL	$\rho = 0.783$ over 12 models	–

6.1 Comparison with Known Dynamics

YC-Bench (He et al., 2026) is a startup-management environment whose transition function is implemented in Python. We replace that function completely with LLM predictions while preserving the task and interface (Section E.1). For fixed action sequences, the LLM predicts the correct direction of the 90-day funds change in 23 of 24 cases with 3.91% mean relative error. When eight agents interact with each simulator, their ranking under LLM dynamics correlates with a coded run at $\rho = 0.857$; two independent coded runs correlate with each other at $\rho = 0.738$.

6.2 Validation Against External Evidence

Without a reference real-world simulator, we compare predictions with plausibility judgments and documented outcomes.

College Career Planning: transition plausibility. Across 413 predicted transitions, three LLM judges rate 89.0–94.1% of direction changes and 86.2–92.2% of magnitudes as plausible. On their shared subset, two humans agree on 95.4% of direction labels, 91.4% of magnitude labels, and 44.4% of trajectory-coherence labels (Section B.6). These non-random subsets measure perceived plausibility, not agreement with real student outcomes.

Startup Strategy: historical replay. Across three-seed replays of 19 companies, the simulator matches 99 of 106 reached financing decisions and 13 of 18 company-status outcomes; matched financings have median errors of $2.0\times$ in amount and $2.36\times$ in post-money valuation. To reduce contamination, names are removed before inference, and we evaluate only time steps after the dynamics model's knowledge cutoff; for these historical replays, we use the older GPT-5.1 model with medium reasoning and no retrieval (Section C.4). Some replay packages informed development, so the results are not fully held out.

NFL Team Management: historical replay. Replaying five coaching tenures yields a mean absolute error of 3.30 wins and title-count rank correlation $\rho = 0.825$. Betting totals achieve a mean

absolute error of 2.27 wins (Sharpe, 2026; nfelo, 2026) but forecast from each season's real roster; one pooled diagnostic informed our final prompt, so development and evaluation are not fully independent (Section D.5).

6.3 Sensitivity to the Dynamics LLM

LLM judges can favor outputs from their own model family (Zheng et al., 2023), so we test analogous dynamics-model bias by replacing GPT-5.5, which supplies College Career Planning and NFL Team Management dynamics, with Claude Opus 5. Holding the prompt, engine, agents, initial states, seeds, and random draws fixed, we find no evidence of such bias in NFL Team Management rankings (Section D.4).

7 Related Work

Strategic decision-making. Recent benchmarks evaluate LLMs on long-horizon decisions in simulated economic and management settings, including Vending-Bench, EcoGym, YC-Bench, CEO-Bench, CoffeeBench, MerchantBench, and FM-Bench (Backlund & Petersson, 2025; Hu et al., 2026; He et al., 2026; Chen et al., 2026; Sugiura et al., 2026; Shi et al., 2026; Wang et al., 2026a). These domain-specific benchmarks use programmatic simulators; we address settings where important effects resist fixed rules.

LLM-based simulation and environment construction. LLMs have simulated social interactions and users and modeled environment dynamics (Park et al., 2023; Zhou et al., 2023; Piatti et al., 2024; Yao et al., 2024; Schmidgall et al., 2024; Saracay et al., 2026; Fronsdal et al., 2025). In the especially relevant Concordia, an LLM game master interprets actions, determines consequences, and generates observations (Vezhnevets et al., 2023; 2025). LLMs have also served as world models for planning (Hao et al., 2023; Wong et al., 2023; Wang et al., 2024b; Zuo et al., 2026; Li et al., 2026) and helped construct simulators (Wang et al., 2023; Tang et al., 2024; Sun et al., 2024; Zhang et al., 2025; Wang et al., 2026b; Kaputa et al., 2026; Hu et al., 2025a). We combine human–LLM environment specification, LLM predictions for difficult effects, and code for exact mechanics.

Evaluator and simulator validity. Prior work validates LLM evaluators and simulators through human agreement, state prediction, or qualitative behavior (Zheng et al., 2023; Li et al., 2023; Shankar et al., 2024; Wang et al., 2024a;b; Park et al., 2024; Hu et al., 2025b; Meshi et al., 2026; Anthis et al., 2025). We also test ranking preservation against coded dynamics, agreement with historical outcomes, and transition plausibility through human and LLM judgments.

8 Discussion & Limitations

LLM-as-Environment combines skill-guided construction, LLM dynamics, and code. Validation against coded dynamics, historical outcomes, and plausibility judgments supports its use, but FuzzyDecisionBench leaves room for improvement. Like all simulators, its fidelity depends on design and validation choices; stronger models and domain expertise should help.

AI use statement

As part of our method, we used LLMs to help construct and calibrate the simulation environments, generate initial states, predict transition dynamics and policy actions within the environment, map documented historical decisions into simulator actions, search for simulator exploits, and evaluate a subset of simulated transitions and trajectories. We also used AI to assist with coding and debugging, running experiments, related work search and retrieval, drafting sections, and polishing the write-up. Sections 3 and 5 and the appendix describe the methodological uses of LLMs and the checks applied to their outputs, including programmatic validation, comparison with historical observations, and human review. The authors reviewed all AI-assisted work and take responsibility for the final text, claims, code, and artifacts.

Ethics statement

FuzzyDecisionBench evaluates models in simulated career, business, and sports decision settings. The environments necessarily encode modeling assumptions about objectives, state, and action effects, and their outputs should not be interpreted as real-world advice for individuals or organizations. Errors or biases in these assumptions could produce misleading recommendations or reinforce existing biases if the simulations were used beyond their intended evaluation setting. We therefore report validation results, limitations, and known data defects.

Reproducibility statement

Section 5.1 describes the evaluation settings, policy models, reference policies, and comparisons. The appendix documents transition procedures, initial-state construction, prompt versions, model settings, seeds, scoring, and validation protocols, including known exceptions and limitations. We release the state and action specifications, frozen dynamics prompts, and scoring and evaluation code needed to reproduce the experiments.

References

Jacy Reese Anthis, Ryan Liu, Sean M. Richardson, Austin C. Kozlowski, Bernard Koch, Erik Brynjolfsson, James Evans, and Michael S. Bernstein. Position: LLM social simulations are a promising research method. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 81005–81034, 2025. URL https://proceedings.mlr.press/v267/anthis25a.html.

Axel Backlund and Lukas Petersson. Vending-bench: A benchmark for long-term coherence of autonomous agents, 2025.

Haozhe Chen, Karthik Narasimhan, and Zhuang Liu. CEO-Bench: Can agents play the long game?, 2026.

Kai Fronsdal, Isha Gupta, Abhay Sheshadri, Jonathan Michala, Stephen McAleer, Rowan Wang, Sara Price, and Sam Bowman. Petri: Parallel exploration of risky interactions, 2025. URL https://github.com/meridianlabs-ai/inspect_petri.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model, 2023.

Muyu He, Adit Jain, Anand Kumar, Vincent Tu, Soumyadeep Bakshi, Sachin Patro, and Nazneen Rajani. YC-Bench: Benchmarking ai agents for long-term planning and consistent execution, 2026.

Erzhen Hu, Yanhe Chen, Mingyi Li, Vrushank Phadnis, Pingmei Xu, Xun Qian, Alex Olwal, David Kim, Seongkook Heo, and Ruofei Du. DialogLab: Authoring, simulating, and testing dynamic human–ai group conversations. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, 2025a. doi: 10.1145/3746059.3747696.

Tiancheng Hu, Joachim Baumann, Lorenzo Lupo, Nigel Collier, Dirk Hovy, and Paul Röttger. SimBench: Benchmarking the ability of large language models to simulate human behaviors, 2025b. URL https://arxiv.org/abs/2510.17516.

Xavier Hu, Jinxiang Xia, Shengze Xu, Kangqi Song, Yishuo Yuan, Guibin Zhang, Jincheng Ren, Boyu Feng, Li Lu, Tieyong Zeng, et al. EcoGym: Evaluating llms for long-horizon plan-and-execute in interactive economies, 2026.

Zoe Kaputa, Anika Rajaram, Vryan Feliciano, Zhuoyue Lyu, Maneesh Agrawala, and Hariharan Subramonyam. SimStep: Human-in-the-loop authoring of interactive educational simulations through task-level abstractions. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, 2026. doi: 10.1145/3772318.3791514.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following models. GitHub repository, 2023. URL https://github.com/tatsu-lab/alpaca_eval.

Yixia Li, Hongru Wang, Peng Lai, Zhiwen Ruan, He Zhu, Youxin Zhu, Ganlong Zhao, Minda Hu, Yun Chen, Sibei Yang, Peng Li, Jeff Z. Pan, Jia Pan, Guanhua Chen, Yang Liu, and Guanbin Li. Bridging the agent-world gap: Text world models for llm-based agents, 2026.

Lennart Luettgau, Vanessa Cheung, Magda Dubois, Keno Juechems, Jessica Bergs, Luke Symes, Henry Davidson, Bessie O'Dell, Hannah Rose Kirk, Max Rollwage, and Christopher Summerfield. People readily follow personal advice from AI but it does not improve their well-being, 2025. URL https://arxiv.org/abs/2511.15352.

Ofer Meshi, Krisztian Balog, Sally Goldman, Avi Caciularu, Guy Tennenholtz, Jihwan Jeong, Amir Globerson, and Craig Boutilier. ConvApparel: A benchmark dataset and validation framework for user simulators in conversational recommenders, 2026. URL https://arxiv.org/abs/2602.16938.

nfelo. NFL season win totals: Implied ratings, odds, and results. https://www.nfeloapp.com/nfl-power-ratings/nfl-win-totals/, 2026. Historical preseason win-total lines, 2003–2026. Accessed 2026-08-28.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *ACM Symposium on User Interface Software and Technology*, 2023.

Joon Sung Park, Carolyn Q. Zou, Jonne Kamphorst, Niles Egan, Aaron Shaw, Benjamin Mako Hill, Carrie Cai, Meredith Ringel Morris, Percy Liang, Robb Willer, and Michael S. Bernstein. LLM agents grounded in self-reports enable general-purpose simulation of individuals, 2024. URL https://arxiv.org/abs/2411.10109.

Giorgio Piatti, Zhijing Jin, Max Kleiman-Weiner, Bernhard Schölkopf, Mrinmaya Sachan, and Rada Mihalcea. Cooperate or collapse: Emergence of sustainable cooperation in a society of LLM agents, 2024. URL https://arxiv.org/abs/2404.16698.

Irena Saracay, Ludwig Schmidt, and Carlos Guestrin. Beyond expert users: Agents should help users construct preferences, not just elicit them, 2026. URL https://arxiv.org/abs/2606.30863.

Samuel Schmidgall, Rojin Ziaei, Carl Harris, Eduardo Reis, Jeffrey Jopling, and Michael Moor. AgentClinic: A multimodal agent benchmark to evaluate AI in simulated clinical environments, 2024. URL https://arxiv.org/abs/2405.07960. Revised May 2025.

Shreya Shankar, J. D. Zamfirescu-Pereira, Björn Hartmann, Aditya G. Parameswaran, and Ian Arawjo. Who validates the validators? aligning LLM-assisted evaluation of LLM outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, 2024. URL https://arxiv.org/abs/2404.12272.

Lee Sharpe. nfldata: Open NFL data. https://github.com/nflverse/nfldata, 2026. Season standings (data/standings.csv) and preseason win totals (data/win_totals.csv). Accessed 2026-08-28.

Qiming Shi, Yulong Tao, Linbo Jin, Zhaolu Kang, Yibo Dou, Jiawen Zhu, Tianjun Pan, Shaokang Fu, Chengyu Wang, Siyue Li, Yaping Cheng, Di Weng, and Chengfu Huo. MerchantBench: Benchmarking llm agents for long-term coherence in e-commerce operations, 2026.

Issa Sugiura, Daichi Hattori, Kazuo Araragi, Keita Ogawa, Shota Onose, Taro Makino, Teppei Usuki, and Takashi Ishida. CoffeeBench: Benchmarking long-horizon llm agents in heterogeneous multi-agent economies, 2026.

Fan-Yun Sun, S. I. Harini, Angela Yi, Yihan Zhou, Alex Zook, Jonathan Tremblay, Logan Cross, Jiajun Wu, and Nick Haber. FactorSim: Generative simulation via factorized representation. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2409.17652.

Hao Tang, Darren Key, and Kevin Ellis. WorldCoder, a model-based llm agent: Building world models by writing code and interacting with the environment, 2024.

Alexander Sasha Vezhnevets, John P. Agapiou, Avia Aharon, Ron Ziv, Jayd Matyas, Edgar A. Duéñez-Guzmán, William A. Cunningham, Simon Osindero, Danny Karmon, and Joel Z. Leibo. Generative agent-based modeling with actions grounded in physical, social, or digital space using Concordia, 2023. URL https://arxiv.org/abs/2312.03664.

Alexander Sasha Vezhnevets, Jayd Matyas, Logan Cross, Davide Paglieri, Minsuk Chang, William A. Cunningham, Simon Osindero, William S. Isaac, and Joel Z. Leibo. Multi-actor generative artificial intelligence as a game engine, 2025. URL https://arxiv.org/abs/2507.08892.

Ruiyi Wang, Haofei Yu, Wenxin Zhang, Zhengyang Qi, Maarten Sap, Graham Neubig, Yonatan Bisk, and Hao Zhu. SOTOPIA-π: Interactive learning of socially intelligent language agents, 2024a. URL https://arxiv.org/abs/2403.08715.

Ruoyao Wang, Graham Todd, Xingdi Yuan, Ziang Xiao, Marc-Alexandre Côté, and Peter Jansen. ByteSized32: A corpus and challenge task for generating task-specific world models expressed as text games. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13455–13471, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.830. URL https://aclanthology.org/2023.emnlp-main.830/.

Ruoyao Wang, Graham Todd, Ziang Xiao, Xingdi Yuan, Marc-Alexandre Côté, Peter Clark, and Peter Jansen. Can language models serve as text-based world simulators?, 2024b.

Tianyou Wang, Chongyang Gao, Kezhen Chen, Dong Chen, Yinghao He, Donghan Li, Wangcheng Xu, Hongjiu Zhang, and Chi Li. FM-Bench: A benchmark for long-horizon management with competing agents, 2026a.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning, 2026b. URL https://arxiv.org/abs/2602.10090.

Lionel Wong, Gabriel Grand, Alexander K. Lew, Noah D. Goodman, Vikash K. Mansinghka, Jacob Andreas, and Joshua B. Tenenbaum. From word models to world models: Translating from natural language to the probabilistic language of thought, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024.

Jiayi Zhang, Yiran Peng, Fanqi Kong, Cheng Yang, Yifan Wu, Zhaoyang Yu, Jinyu Xiang, Jianhao Ruan, Jinlin Wang, Maojia Song, HongZhang Liu, Xiangru Tang, Bang Liu, Chenglin Wu, and Yuyu Luo. AutoEnv: Automated environments for measuring cross-environment agent learning, 2025. URL https://arxiv.org/abs/2511.19304.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. SOTOPIA: Interactive evaluation for social intelligence in language agents, 2023. URL https://arxiv.org/abs/2310.11667.

Yuxin Zuo, Zikai Xiao, Li Sheng, Fei Huang, Jianhong Tu, Yuxuan Liu, Tianyi Tang, Xiaomeng Hu, Yang Su, Qingfeng Lan, Yantao Liu, Qin Zhu, Yinger Zhang, Bowen Yu, Haiquan Zhao, Haiyang Xu, Jianxin Yang, Jiayang Cheng, Junyang Wang, Lianghao Deng, Mingfeng Xue, Tianyi Bai, Yang Fan, Yubo Ma, Yucheng Li, Zeyu Cui, Zhihai Wang, Zhihui Xie, Zhuorui Ye, An Yang, Dayiheng Liu, Jingren Zhou, and Ning Ding. Qwen-AgentWorld: Language world models for general agents, 2026.

A Shared Implementation and Reproducibility Details

A.1 Transition implementation

The engine validates an action against the action schema and the current state before it calls the dynamics LLM. It then applies any code-defined preprocessing and constructs the LLM input from the state, action, relevant history, derived quantities, and definitions of the state variables and actions. The dynamics LLM returns a structured prediction. Some transitions use more than one LLM call because they resolve separate events. The engine validates each prediction, rejects changes to fields that the LLM does not own, records resolved events, applies the remaining code-defined rules, and recomputes derived quantities and utility when required.

Table 4 summarizes the update mechanisms. The categories describe implementation roles rather than disjoint transition types: resolving an event can require an LLM judgment followed by a code-defined update. The YC-Bench study replaces the original transition function more directly and is described in Section E.1.

Table 4: Ownership of updates in the three FuzzyDecisionBench environments.

Mechanism	Responsibility	Examples
Dynamics LLM	Predict changes that require qualitative judgment	Skill growth, customer response, fatigue, and scheme fit
Programmatic engine	Apply explicit rules, arithmetic, constraints, and seeded draws	Time, budgets, eligibility, deadlines, and salary-cap accounting
Event resolution	Turn an LLM judgment into a durable record	Application decisions, offers, contracts, and terminal outcomes
Derived quantities	Compute values from other state variables	Runway, payroll, well-being, and salary-cap space

A.2 Frozen experiment settings

Table 5 records the settings supported by the paper repository. The reported College Career Planning comparison evaluates each model–cell pair. The reported NFL Team Management comparison averages seeds 0–2. The Startup Strategy comparison evaluates each model–state pair across its five initial states.

Table 5: Frozen settings for the reported evaluations. "Common" denotes the 11 models reported for all three FuzzyDecisionBench environments in Table 2.

Study	Dynamics	Evaluated runs	Reported aggregate
College Career Planning	GPT-5.5, temperature 0, prompt v15	Seven selected objective–state cells; 11 common models and 15 models in the broader result set	Normalize within each cell using π_{lower} and π_{upper}, then average cells
Startup Strategy	GPT-5.6 Sol, medium reasoning	Five initial states; 11 common models	Normalize within each state using π_{lower} and π_{upper}, then average states
NFL Team Management	GPT-5.5, temperature 0, prompt v11	Five scored states and seeds 0–2; 11 common models and 14 models in the per-state result set	Sum mean expected titles over the five scored states, then normalize using aggregate reference totals
YC-Bench replacement	GPT-5.5, temperature 0, medium reasoning	Eight policy models and seeds 1–3	Mean funds across seeds, followed by model ranking

A.3 Reference-policy searches and scoring

For each selected College Career Planning cell, π_{lower} follows a hand-coded schedule. The corresponding π_{upper} trajectory is the best result from five independent GPT-6 Astra Codex searches. The searches can read the environment code and dynamics prompt, inspect simulated outcomes, and revise candidate trajectories. They use five search emphases: warm-start refinement, career-gate pursuit, well-being-first optimization, mechanics and timing analysis, and a structurally different hybrid. The paper repository does not record a comparable call or token budget for these searches.

In NFL Team Management, π_{lower} makes no optional roster or coordinator changes, spends nothing on those actions, and uses balanced schemes with a trust-building, competitive posture. The π_{upper} search selects one trajectory for each scored state and can inspect the engine's seeded random draws. The selected trajectories therefore benefit from search and information unavailable to evaluated models.

In Startup Strategy, π_{lower} follows a deterministic founder policy: it develops one focused product, moves from private beta to general availability, updates reliability and onboarding every three months, maintains two sales pursuits, and raises the first available priced round when runway falls below nine months. It does not hire, cut staff, pivot, borrow, form partnerships, acquire another company, or sell the company. For each initial state, π_{upper} is the best of three privileged GPT-6 Astra coding-agent searches; the wholesale-operations state received three additional searches because the first reference was below an evaluated policy. The searches can inspect the environment implementation, dynamics prompt, private simulator state, and outcomes from candidate trajectories. The reference policies are empirical comparison points, not performance bounds.

A.4 Aggregate trajectory diagnostics

Figure 4 summarizes outcome variation across the evaluated trajectories.

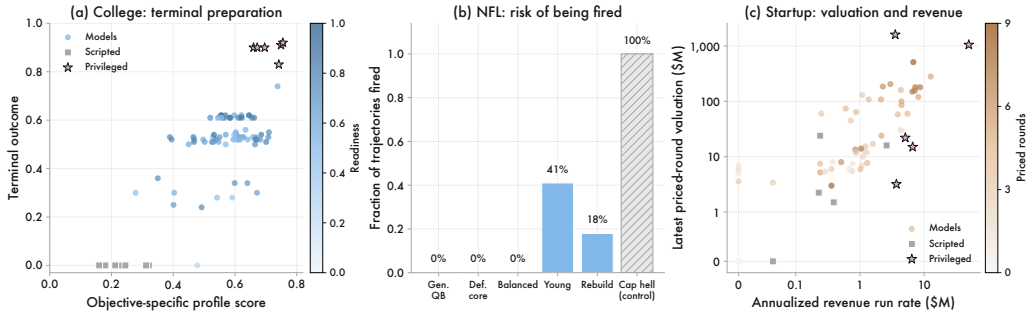

Figure 4: **Aggregate trajectory diagnostics.** (a) Profile score, terminal outcome, and readiness for 77 College Career Planning model–objective trajectories and their references. (b) Firing rates by NFL Team Management initial state across 13 models and three seeds; cap hell is the forced-firing control. (c) Revenue, latest priced-round valuation, and financing count for 55 Startup Strategy model–state trajectories and their references; both axes use symmetric-log scaling in millions of dollars. The panels are descriptive and do not identify the causal effect of any action.

B College Career Planning

B.1 Overview

Decision problem. The agent acts as an undergraduate through 15 quarters (first-year fall through fourth-year spring, including three summers). Each quarter it divides 148 weekly hours (20 omitted for basic life maintenance) among coursework, research, projects, competitions, sleep, and other activities, and it may apply for, accept, or decline positions.

State, actions, and dynamics. The dynamics LLM updates 47 of the 78 state variables, including skills, career-track profiles, relationships, and health. Aptitudes and market conditions stay fixed

after initialization. The code engine maintains enrollment status, the declared major, and the lists of offers, positions, accomplishments, and test scores. Beyond these updates, the dynamics LLM decides whether each application, submission, and competition entry succeeds and at what tier. The engine records each successful application as an offer with a deadline, and it enforces rules on timing and commitments. For example, full-time recruiting happens only in the last four quarters, and the student can hold at most one full-time job and one graduate or professional school placement. The engine then sets a career outcome, a score in $[0, 1]$ that depends on the tier of the offer, only when the student accepts a full-time job offer or a school admission.

Initial states and utility. We have seven evaluation instances, each representing a single career track: technology, finance, consulting, academic research, medicine, law, and creative work. Here, we define utility as the average of the student's career outcome and the student's well-being. The well-being term averages the student's mean and final well-being, so a late recovery cannot erase earlier quarters of poor well-being (Section B.3). To construct initial states, we define nine student archetypes, such as an aspiring researcher and a pre-med student. For each archetype, a seeded Gaussian copula samples correlated aptitudes and early skills. An LLM then fills in the remaining variables to match these values and a short description of the student's circumstances, such as working part-time. We evaluate each objective on one initial state, chosen from seven candidates after inspecting privileged-search trajectories to favor states that lead to distinct strategies (Section B.4).

B.2 State and actions

The state groups and action families are listed in Tables 6 and 7. Applications and other selective events use separate resolution steps. Ordinary dynamics predictions cannot overwrite terminal outcomes, offers, accomplishments, test records, fixed aptitudes, or enrollment status. The engine records resolved events and enforces the corresponding rules.

Table 6: Main groups of College Career Planning state variables.

Group	Examples
Academics	GPA, degree progress, course rigor, and major
Skills and aptitudes	Technical skills, reasoning, communication, collaboration, leadership, and fixed aptitudes
Career preparation	Career profiles and application readiness for seven career paths
Achievements and ventures	Projects, publications, test results, accomplishments, and startup progress
Relationships and opportunities	Faculty and industry relationships, peer networks, and pending offers
Well-being and circumstances	Physical and emotional health, stress, social connection, finances, and external conditions
Commitments and status	Enrollment, active positions, and accepted placements
Terminal outcomes	Career, school-admission, and startup outcomes

Table 7: College Career Planning action families.

Family	Examples
Time allocation	Coursework, research, career work, projects, preparation, networking, volunteering, exercise, sleep, and social time
Applications and submissions	Apply for jobs or programs, submit publications or creative work, enter competitions, and take tests
Opportunity decisions	Accept, decline, or negotiate an offer
Ventures and organizations	Launch or close a startup, found an organization, and form a team
Academic and enrollment decisions	Change major, study abroad, take leave, return, leave a position, or drop out

The policy allocates 148 hours per week after a fixed allowance for basic daily tasks. Sleep counts toward this budget. Discrete choices also consume time; the engine converts their quarterly effort to an equivalent weekly load. Code checks the total allocation, application windows, eligibility, placement limits, offer deadlines, and graduation requirements.

B.3 Utility and initial states

The seven objectives cover technology, finance, consulting, academic research, medicine, law, and creative careers. Each objective gives equal weight to its terminal career outcome and well-being:

$$U = 0.50\, U_{\text{career}} + 0.50\, U_{\text{well-being}}.$$

Well-being combines the mean over the episode and the terminal value:

$$U_{\text{well-being}} = 0.5\, \overline{W} + 0.5\, W(s_H).$$

At each recorded state, W averages emotional health, one minus stress, physical health, and the mean of self-efficacy and social connectedness. The intended episode mean includes the terminal state.

We define nine student archetypes and generate ten candidate states for each. A seeded Gaussian-copula sampler produces correlated aptitudes and early skills from archetype-specific truncated normal distributions. An LLM fills the remaining fields from these values, the state-variable definitions, and one or two descriptions of the student's circumstances. A second pass checks consistency and diversity within each archetype. After both passes, the initializer restores the sampled numerical values so the LLM cannot change them. Code then checks ranges, fills required defaults, initializes relationships and earned outcomes, and flags near-duplicate states.

Development trajectories use initial-state indices 8–9. The main benchmark cells use selected indices from 0–6. This split separates the benchmark trajectories from the initial states used to tune or diagnose the dynamics, but it does not show that every case in the plausibility study was unused during development.

B.4 Evaluation cells and additional results

The evaluation pairs each objective with one selected initial state. The cells were selected after inspection of the π_{upper} trajectories, with preference for cells that produced distinct strategies. This is a post-hoc selection based on reference trajectories. Table 8 lists the selected cells.

Table 8: The seven College Career Planning cells used in the reported evaluation.

Objective	Initial-state index	π_{upper} utility
Academic researcher	0	0.890
Consulting career	5	0.846
Creative career	4	0.928
Finance career	1	0.833
Pre-law	6	0.916
Pre-med	3	0.871
Technology optimizer	4	0.905

The common comparison in Table 2 contains 11 models and therefore 77 policy trajectories. The broader College Career Planning result set in Table 9 contains 15 models and therefore 105 policy trajectories. Each row averages raw utility over seven trajectories and averages normalized utility after normalizing within each objective–state cell. The mean raw utilities of π_{lower} and π_{upper} are 0.386 and 0.884.

Table 9: Broader College Career Planning results. Each row averages the seven selected objective–state cells.

Model	Raw utility	Normalized utility
GPT-6 Astra	0.663	55.6%
Claude Opus 5.5	0.640	51.0%
GPT-5.6 Sol	0.638	50.6%
Gemini 3.8 Flash	0.629	48.8%
Kimi K3	0.629	48.8%
Claude Opus 5	0.629	48.6%
Claude Fable 5.1	0.624	47.8%
Gemini 3.1 Pro	0.622	47.5%
Claude Fable 5	0.617	46.4%
GLM-5.3	0.610	44.8%
Gemini 3.7 Flash	0.599	42.9%
DeepSeek V4 Pro	0.584	39.8%
GPT-5.6 Luna	0.582	39.3%
Qwen3.8-Max	0.583	39.3%
Claude Sonnet 5	0.526	28.0%
π_{lower}	0.386	0.0%
π_{upper}	0.884	100.0%

B.5 Planning-horizon ablation

We test whether College Career Planning rewards long-horizon planning rather than repeated local action selection. The experiment uses GPT-6 Astra with medium reasoning for all policies and the same seven objective–state pairs, 15-quarter horizon, observations, utility functions, action space, and action history as the main evaluation. GPT-5.5 with medium reasoning and temperature 0 provides dynamics. The policies do not see the dynamics prompt, engine code, hidden state, or alternative rollouts.

We compare three policies. The *myopic* policy selects the action that maximizes expected utility after the next transition and ignores later effects. The *direct full-horizon* policy receives the full remaining horizon and selects each action without an explicit persistent plan. The *persistent full-horizon* policy also maintains milestones, guardrails, contingencies, and beliefs, and may replace its plan when an observed trigger fires. Each policy advances the environment once per quarter; none performs search or calls the dynamics model on hypothetical actions.

Table 10: **Planning-horizon ablation in College Career Planning.** Entries for the seven evaluation cells are raw utility. Normalized means use the same cell-specific reference policies as the main evaluation.

Objective	Myopic	Direct full horizon	Persistent full horizon
Academic research	0.453	0.646	0.660
Consulting	0.434	0.640	0.613
Creative career	0.467	0.651	0.670
Finance	0.445	0.601	0.586
Law	0.471	0.725	0.724
Medicine	0.469	0.662	0.671
Technology	0.460	0.709	0.702
Raw mean	0.457	0.662	0.661
Normalized mean	14.2%	55.3%	55.1%

Both full-horizon policies beat the myopic policy in all seven cells. For direct full-horizon versus myopic, the mean raw-utility difference is 0.205, the mean normalized difference is 41.1 percentage points, and a two-sided exact sign test gives $p = 0.0156$. The myopic policy ends with a zero target career or school outcome in every cell, whereas both full-horizon policies obtain a target outcome between 0.52 and 0.62 in every cell. Thus, the difference comes from actions that build toward delayed placements rather than from a uniform change in immediate well-being.

The persistent plan does not improve on direct full-horizon action selection: its mean raw utility is lower by 0.001, and it wins three of seven cells. It also retains its initial plan in every quarter of every trajectory, despite observed replanning triggers. This ablation supports the claim that the benchmark rewards planning over delayed consequences; it does not show a benefit from this persistent-plan scaffold. Because each cell has one run, the experiment does not estimate variation from remote model inference.

Figure 5 shows how the objective-specific profile score develops in every reported cell for the leading aggregate model and both references. This diagnostic does not include readiness, outcomes, or well-being and therefore is not a substitute for utility.

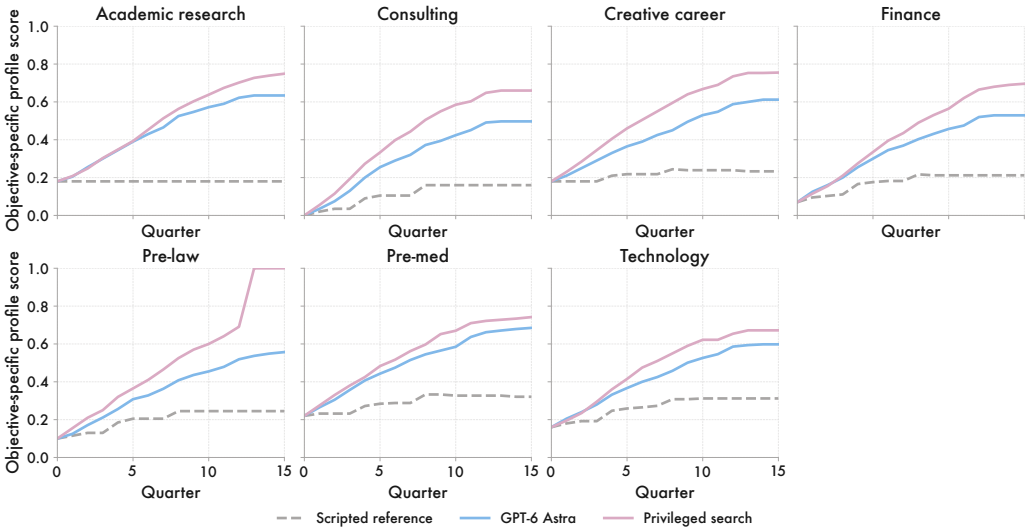

Figure 5: **College Career Planning trajectory atlas.** Objective-specific profile scores over 15 quarters for the scripted reference, GPT-6 Astra, and privileged search in the seven reported objective–state cells. GPT-6 Astra is shown because it has the highest aggregate normalized utility; the privileged line is the best of five searches in each cell.

B.6 Validation procedures and provenance

Development tests. During development, the authors use hand-authored cases, adversarial actions, comparisons between actions from the same state, and complete action sequences. The tests cover selective outcomes, repeated attempts, networking, workload, well-being, penalties for withdrawing from commitments, and the effect of skills and demonstrated achievements on later opportunities. These tests can lead to changes in the environment and are not validation evidence.

Reported plausibility study. The broader case bank contains 639 single-step cases and 44 complete action sequences. The reported study evaluates 413 transition predictions: 53 hand-authored cases and 360 transitions sampled from 90 policy trajectories. GPT-5.4-mini, Gemini 3 Flash, and Claude Haiku 4.5 generate the 90 trajectories across 15 utility settings and two initial states. Sampling is stratified by policy model, utility setting, and initial state. Code checks action feasibility, output format, fixed variables, permitted changes, and variables constrained to move in only one direction.

Claude Opus 4.6, Gemini 3.1 Pro, and GPT-5.5 judge the direction and magnitude of selected changes. They receive the current state, action, predicted changes, case description, and state-variable definitions, but not the expected answer for the case. They also judge 102 complete trajectories for coherence. The first three rows of Table 11 report their assessments; the final row reports agreement between the two human raters on their shared cases. These scores measure plausibility to the judges, not agreement with observed student outcomes.

Table 11: Plausibility assessments for College Career Planning. The LLM rows report judgments of 413 transition predictions and 102 trajectories. The human row reports inter-rater agreement on the shared 24 predictions (175 factor-level decisions) and nine trajectories.

Judge	Direction	Magnitude	Coherence
Claude Opus 4.6	94.1%	87.8%	97.1%
Gemini 3.1 Pro	93.8%	92.2%	92.2%
GPT-5.5	89.0%	86.2%	95.1%
Human 1–Human 2	95.4%	91.4%	44.4%

Two human raters evaluate partially overlapping strategic subsets from the same study. The original rater evaluates 61 predictions and 12 trajectories, supplying 610 factor-level labels and 12 trajectory-coherence labels; the second evaluates 60 predictions and 12 trajectories, supplying 597 factor-level labels and 12 trajectory-coherence labels. Each subset is balanced across career paths and policy models, with selection within groups prioritizing disagreement among the LLM judges. The original rater's mean agreement with the three LLM judges is 90.9% for direction, 84.7% for magnitude, and 75.0% for coherence.

The two human subsets share 24 predictions, comprising 175 common factor-level decisions, and nine trajectories. On this overlap, human–human raw agreement is 95.4% for direction, 91.4% for magnitude, and 44.4% for trajectory coherence; Cohen's κ is 0.185, 0.089, and 0.000, respectively. The low factor-label κ values reflect the strong prevalence of `correct` and `about_right` labels despite high raw agreement. Because the subsets are non-random and only partially overlap, these figures estimate inter-rater reliability on the shared cases rather than population-level accuracy.

C Startup Strategy

C.1 Overview

Decision problem. The agent acts as the founder and CEO of a software startup for 36 months, with one step per month. Each month, it sets a standing operating plan, including spending by function and hiring targets, which persists until changed. It can also revise the company's pitch, launch products, set customer segments and prices, and pursue large contracts. It can hire executives or cut staff, raise equity or debt, form partnerships, acquire other companies, sell the company, or wind it down. The agent must grow recurring revenue and raise capital before cash runs out, since a company whose cash stays negative for two months becomes insolvent.

State, actions, and dynamics. The state has 25 structured variables with about 190 fields, covering the founders and their network, the pitch, products, customers, team, financials, the cap table, fundraising, market conditions, and deals. The dynamics LLM's judgments update 18 of these variables, and the engine owns the other 7, including the financials, the cap table, and the operating plan. Many action arguments are free text, such as the pitch or the terms proposed to a customer, but the agent can only state intent and cannot set realized outcomes such as customers, revenue, hires, or valuations. Each month, the dynamics LLM makes three judgments in sequence: execution, such as product progress, reliability, and hiring; the commercial response, such as new customers, renewals, and churn; and the capital-market response, such as financing offers and acquisition interest. Each judgment sees the results of the earlier ones, and the capital judgment also sees the engine's preview of the month's finances. The engine then commits the month. It closes transactions, updates revenue, payroll, cash, debt, and the cap table, and advances the market.

Initial states and utility. Utility is terminal company value. A completed sale scores its realized acquisition price, an insolvent company scores zero, and an orderly wind-down scores its nonnegative residual cash after senior claims. Otherwise, the score is the pre-money valuation of the latest priced round plus ten times the change in sustained repeat annual revenue since that round, floored at zero; if the company never raised a priced round, the score is ten times terminal sustained repeat annual revenue. Sustained repeat annual revenue is the lowest annualized contracted-recurring, retained usage, and retained marketplace take-rate revenue over the final six months. Founder dilution

does not affect the score. We hand-design five fictional openings so that the default playbook of a founder-led pilot followed by a fundraise is wrong or incomplete: an AI research team in an exuberant financing market, a stalled seed-stage company with seven months of runway, a company with a live acquisition offer, an enterprise software company facing long sales cycles, and a small-business lender that needs outside capital in a cautious market. Three openings start with no product or revenue, and two start as operating companies. Code builds each opening into a full state and checks its accounting.

C.2 Environment and opening states

The environment represents a company at monthly resolution for 36 steps. The observation records the founding team and network; pitch and product approach; products and milestones; commercial initiatives, contract pursuits, customer tiers, and revenue; executives, key people, and functional headcount; operating budgets; cash, debt, financing processes, and the cap table; market conditions; partnerships, acquisitions, exit processes, and company status. Engine-private memory retains dynamics judgments needed for continuity, such as inferred customer demand and the primary constraint on each commercial stream.

An action contains persistent operating-plan settings and structural decisions. The model can revise the pitch and approach; create, update, and distribute products; set product goals and customer targets; launch commercial initiatives and pursue contracts; hire or reduce staff; cultivate investors, raise equity or debt, and respond to term sheets; form partnerships; acquire or sell a company; or wind down. Standing plans persist until changed. Bounded initiatives and transaction processes have explicit identifiers and timing.

Utility is terminal company value in U.S. dollars. A completed sale uses its realized acquisition price, an insolvent company receives zero, and an orderly wind-down receives its nonnegative residual cash after senior claims. For an operating company with a priced round, the engine starts from the latest pre-money valuation and adds ten times the change in sustained repeat annual revenue since that round, flooring the result at zero. Without a priced round, terminal value is ten times terminal sustained repeat annual revenue. SAFE, note, grant, and accelerator valuations do not count as priced rounds. The objective does not multiply value by founder ownership, so it measures company value rather than the founders' retained stake.

The dynamics LLM makes typed judgments about execution, commercial response, and capital-market response from the state, action, and relevant history. The engine records management intent, settles contracts and transactions, updates revenue, cost of revenue, payroll, cash, debt, dilution, and the cap table, advances the market and clock, checks accounting constraints, and computes terminal value.

The five openings are fixed full states. Three have no product or revenue. The wholesale-SaaS and accounts-payable cases begin at company ages 14 and 20 months with operating histories. Code recomputes cash, burn, runway, payroll, revenue aggregates, and cap-table invariants before storing each state. Historical replay uses separate states reconstructed from dated evidence; each field distinguishes observed values from assumptions.

Table 12: The five Startup Strategy openings. Runway is computed from opening cash and net burn.

Opening	Cash	Runway	Financing	Initial condition
AI research team	$720k	12.0 mo.	exuberant	idea stage
Wholesale operations SaaS	$520k	7.4 mo.	selective	stalled seed company
Accounts-payable automation	$2.5M	36.2 mo.	active	live $30M offer
Enterprise logistics software	$125k	17.9 mo.	selective	long sales cycle
SMB lending fintech	$65k	11.8 mo.	constrained	regulated market

C.3 Five-state results

The reported comparison crosses the 11 common policy models with the five initial states, for 55 model–state episodes. For each state, we normalize terminal company value using the matched scripted and privileged-search references, then average the five normalized values. Because nor-

malization occurs before averaging, the high-dollar AI-research and accounts-payable states do not dominate the aggregate.

Table 13: Startup Strategy results for each initial state. State columns report terminal company value U in millions of dollars; the final column averages the five state-normalized utilities. Bold marks the best evaluated policy in each column. The reference rows show the raw values used to normalize each state.

Policy	AI research	Wholesale ops	AP acquisition	Ent. logistics	SMB lending	Mean $\widetilde{U}$
Claude Fable 5.1	190.2	19.2	183.1	**19.2**	6.8	**25.1%**
GPT-6 Astra	111.6	**28.2**	70.9	16.6	3.4	21.0%
Claude Opus 5.5	**529.2**	15.5	164.3	5.0	13.8	19.3%
GPT-5.6 Sol	209.8	17.1	**295.0**	0.0	**17.9**	15.7%
Kimi K3	178.0	15.6	144.1	0.0	9.7	8.5%
Gemini 3.8 Flash	45.0	11.4	57.1	6.0	8.5	8.5%
DeepSeek V4 Pro	112.3	8.8	184.9	7.0	0.0	7.9%
GPT-5.6 Luna	67.0	9.2	95.2	6.0	5.2	7.3%
Claude Sonnet 5	133.6	8.1	154.1	0.0	3.6	3.2%
GLM-5.3	60.0	14.5	103.1	0.0	3.0	3.0%
Qwen3.8-Max	74.0	0.0	66.5	0.0	7.5	0.3%
π_{lower}	25.8	2.4	30.3	4.1	0.6	0.0%
π_{upper}	1603.6	71.5	1138.8	28.1	45.4	100.0%

Figure 6 compares financing trajectories across all five opening states. It plots latest priced-round valuation rather than terminal utility so that the timing of financing decisions is visible.

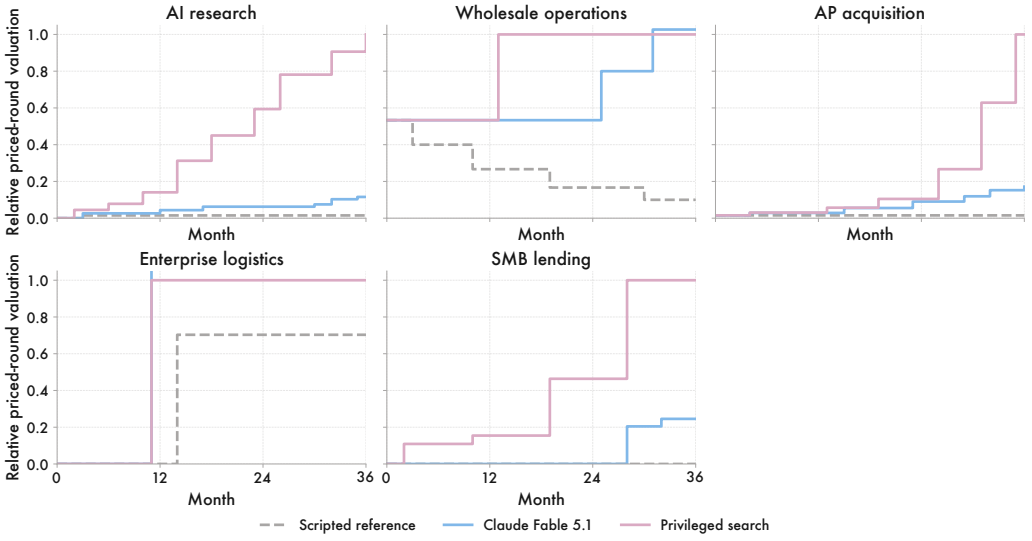

Figure 6: **Startup Strategy trajectory atlas.** Latest priced-round valuation over 36 months for the scripted reference, Claude Fable 5.1, and privileged search in the five opening states. Within each panel, valuation is divided by that state's privileged terminal priced-round valuation. Claude Fable 5.1 is shown because it has the highest mean normalized utility. This plotted valuation is one component of terminal utility.

C.4 Historical conditional-action replay

We construct replay packages for 19 companies from dated public evidence. We assembled the source ledger from dated public records: company announcements and product changelogs; SEC

filings and press releases; founder and company posts on X and LinkedIn; PitchBook; and reporting from TechCrunch, Bloomberg, The Information, Reuters, CNBC, Forbes, and industry outlets. The market calendar draws separately on PitchBook–NVCA, Carta, Crunchbase, Federal Reserve and FRED series, and dated reporting on external shocks. Package manifests list the sources used to reconstruct each opening state, and every scored ground-truth factor records a direct URL and publication date. We use archived pages when the original is no longer live. Each package contains an opening state, a fixed sequence of documented management actions, sparse ground-truth checkpoints, and a dated sequence of exogenous conditions. The exogenous sequence can set financing climate, sector sentiment, and outside market events, but cannot set revenue, customers, hiring, financing outcomes, valuations, transactions, or company status. The simulator advances from its own predicted state without correction to the historical state. The study therefore asks what the environment predicts conditional on the documented actions and exogenous conditions; it does not test whether an autonomous policy would choose those actions.

Company, product, founder, executive, and prior-employer names are replaced with consistent placeholders before inference. For the primary replay, we run each package repeatedly using GPT-5.1 with medium reasoning and no retrieval. To reduce contamination risk, we evaluate predictions only at time steps after the dynamics model's knowledge cutoff. A financing decision is matched within its documented action month or the next six months. Round amounts and post-money valuations are compared only when a round occurs in both the historical record and the simulation. Unavailable or incomparable historical values are omitted rather than scored as zero.

Table 14 pools all reached, scoreable windows across runs. The pooled set includes packages used during environment development, so it is not a held-out estimate.

Table 14: Startup Strategy conditional-action replay. Denominators include only windows reached by a replay with a comparable historical target. Financing amounts and valuations are evaluated only for matched rounds.

Quantity	Result	Unit of analysis
Financing-decision agreement	93.4%	99 of 106 windows
Historical financing recovered	92.5%	86 of 93 positive windows
Spurious financing avoided	100%	13 of 13 negative windows
Median amount error	factor 2.0	77 matched rounds
Median post-money error	factor 2.36	62 matched priced rounds
Status agreement	72.2%	13 of 18 endpoints

For matched rounds, the geometric-mean simulated amount is 0.45 times the historical amount and the geometric-mean simulated post-money valuation is 0.27 times the historical value. These comparisons measure agreement with recorded outcomes under the replay protocol, not the accuracy of autonomous startup forecasts.

We additionally run a separate exploratory replay with GPT-5.6 Sol as the dynamics model. Rather than treating narrow point tolerances as the primary measure of fidelity, Figures 3 and 7 report trajectory direction and signed factor-scale error. Across 57 comparable positive revenue observations, the median simulated-to-historical ratio is 0.20; 15 are within a factor of two and 27 are within a factor of five. The direction of change agrees for 25 of 47 consecutive within-company checkpoint pairs, using a 5% deadband around no change. No simulated revenue checkpoint exceeds its historical comparator by more than a factor of two. Ranges are represented by their geometric midpoint, and historical lower bounds are plotted at the bound, so these summaries are deliberately descriptive rather than pass–fail scores.

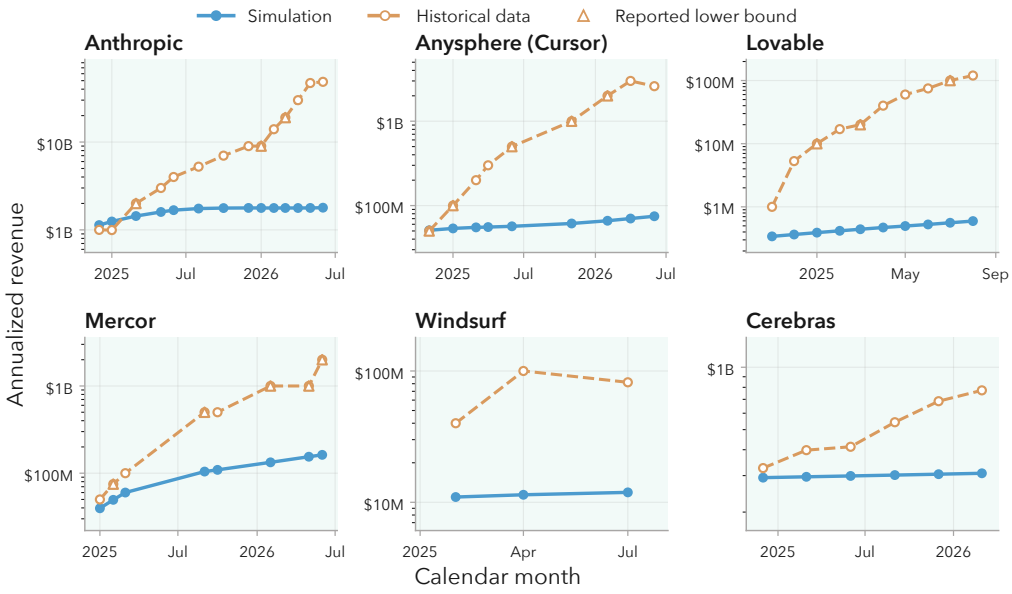

Figure 7: **Simulated and historical revenue trajectories.** The six companies shown have the densest historical coverage and account for 50 of the 57 comparable positive observations. Monthly revenue is annualized, and the vertical axes use logarithmic scales; triangles mark reported lower bounds. Across all ten companies, the median simulated-to-historical ratio is 0.20, with 15 observations within a factor of two and 27 within a factor of five.

In this GPT-5.6 Sol replay, 11 of 24 reached historical financing windows contain a simulated round and 13 do not. Conditional on a matched round, the median simulated-to-historical ratio is 0.57 for amount and 0.48 for post-money valuation; 6 of 11 amounts and 5 of 10 valuations are within a factor of two, while 9 of 11 amounts and 8 of 10 valuations are within a factor of five. Every matched amount and valuation is below twice its historical comparator. For two rounds, the historical action description specifies the amount exactly; these points are marked separately because the amount is not an independent simulation prediction.

D NFL Team Management

D.1 Overview

Decision problem. The agent acts as an NFL head coach for up to ten seasons, with trajectories ending early if the coach gets fired. Each step is one offseason in which the agent sets how aggressively to pursue free agents and trades, chooses a draft strategy, and decides whether to extend, trade, or cut each star player. It can also choose offensive and defensive schemes, fire a coordinator, and decide whether to compete or tank for a higher draft pick. The agent must decide when to spend limited cap space and draft capital as a "win now" move and when to build for later seasons.

State, actions, and dynamics. The dynamics LLM updates 9 of the 36 state variables, including the strength of each of seven position groups, locker-room mentality, and owner trust. The code engine maintains variables like player contracts, the salary cap, and player ages. Each step begins with the engine, which checks that the agent's plan fits under the salary cap, applies its contract decisions, and draws the offseason's signings, trades, and draft picks. The probabilities of these draws depend on the agent's choices and on the quality of that year's free-agent and draft classes. The dynamics LLM then resolves the season from the updated roster. It predicts how each position group changes, the team's regular-season wins, and how far the team advances in the playoffs. If the team reaches the Super Bowl, the LLM predicts its probability of winning. The LLM also decides

whether the owner fires the coach based on wins, owner trust, market size, and consecutive losing seasons. Finally, the engine draws the Super Bowl result from the predicted probability.

Evaluation instances and utility. We define utility as the *expected* number of Super Bowl titles, i.e., the sum of the predicted win probabilities over the seasons in which the team reaches the Super Bowl. Using these probabilities instead of realized titles removes the variance from the engine's random draw of each Super Bowl result. We have five evaluation instances representing different team strengths: a generational quarterback, a generational defense, a balanced contender, a young roster with cap space, and a rebuilding team. We evaluate each instance under three engine seeds, which fix the engine's random draws.

D.2 Environment specification and initial states

The state records the roster and position-group strength, player age, contracts and dead money, salary-cap space, coordinators and schemes, owner trust, market conditions, prior results, and the coach's employment status. The model makes offseason decisions about free-agent signings, trades, the draft, coordinator changes, schemes, spending, and competitive posture. The engine owns contracts, salary-cap accounting, aging, clocks, and seeded event draws. The dynamics LLM judges roster development, coordinator and scheme fit, team quality, owner response, and other transitions not fixed by explicit rules. Each episode lasts at most ten seasons and terminates early if the coach is fired. Utility is the expected number of Super Bowl titles over the episode.

We materialize one state for each of five scored archetypes: a generational-quarterback window, a defensive-cornerstone window, a balanced contender, a young roster with salary-cap space, and a new coach entering a rebuild. Roster strength, age, coordinators, owner trust, and market conditions are fixed or sampled from specified distributions. Sampling uses master seed 20260826 and separate deterministic random streams for each archetype. Before storing a state, the engine recomputes salary-cap space from contracts and dead money and checks the schema.

A sixth state, an aging roster with severe salary-cap constraints, is executed as a negative control. Its reference gap is zero: π_{lower}, π_{upper}, and every evaluated model receive zero expected-title utility under the frozen configuration. The negative control is excluded from reported five-state totals and normalized scores. It is included in the trajectory counts for experiments that executed all six states.

D.3 Evaluation, references, and aggregation

The broader evaluation crosses 14 policy models, six executed states, and seeds 0, 1, and 2, for 252 episodes. The five scored states account for 210 episodes; the negative control accounts for 42. The recorded dataset contains 1,964 simulated seasons, and 73 episodes end early when the coach is fired. The main table reports the 11 models shared by all three FuzzyDecisionBench environments.

For each model, the raw score is computed by averaging expected titles over the three seeds within each scored state and summing the five state means. The π_{lower} and π_{upper} totals are 6.653 and 29.050 expected titles. The normalized score is

$$\widetilde{U}(\pi) = \frac{U(\pi) - 6.653}{29.050 - 6.653},$$

where $U(\pi)$ is the five-state total. The negative control does not enter this calculation. Across the broader set of policy models, the standard deviation across the three seed-level totals ranges from 0.316 to 2.280 expected titles. This is variation across three runs, not a standard error.

D.4 Per-state results and dynamics-model comparison

Figure 8 reports the 14-model result set for each of the five scored states. The generational-quarterback and defensive-cornerstone states account for 85%–94% of each model's five-state total. The figure prints the raw mean over seeds in each cell and uses color to show the fraction of that state's reference interval reached.

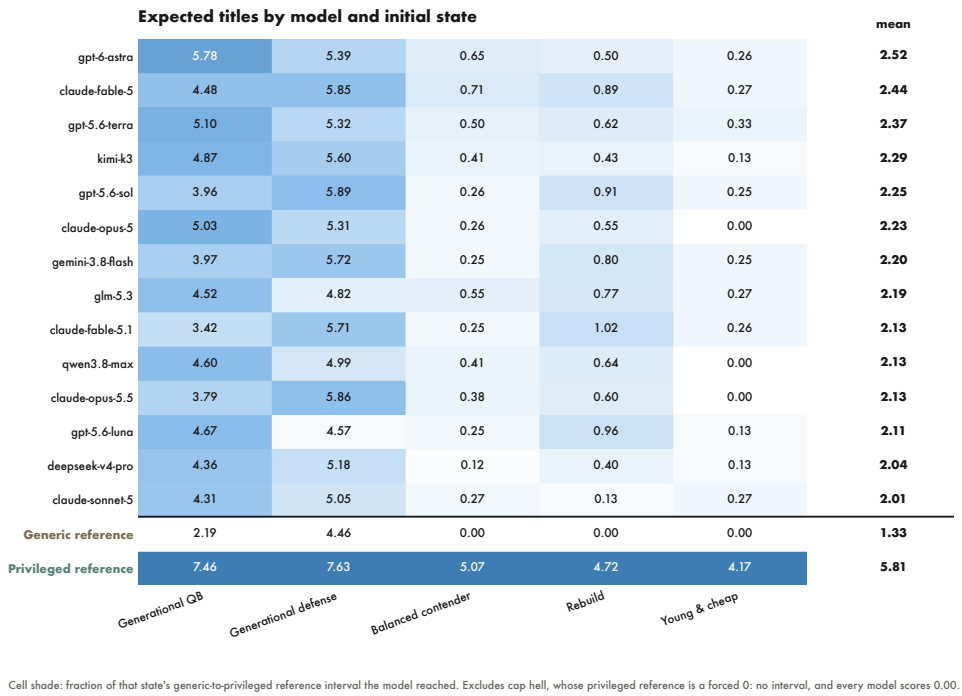

Figure 8: NFL Team Management expected titles by model and scored initial state. Each cell is the mean over seeds 0–2. Color is the fraction of the state-specific interval from π_{lower} to π_{upper}. The negative control is omitted because its reference gap is zero.

Figure 9 shows the seed-0 title trajectories for every executed initial state, including the negative control.

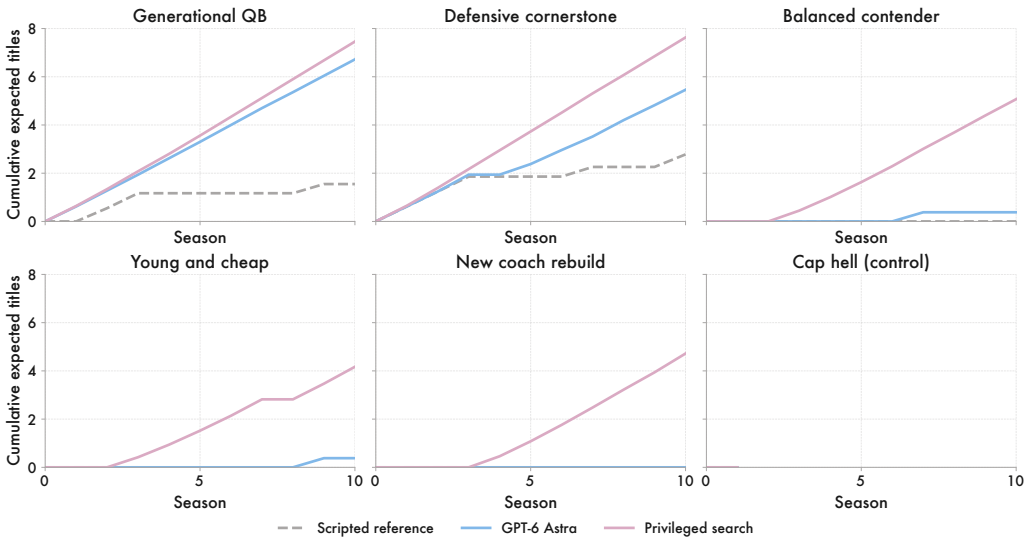

Figure 9: **NFL Team Management trajectory atlas.** Cumulative expected titles by season for the scripted reference, GPT-6 Astra, and privileged search at seed 0. GPT-6 Astra is shown because it has the highest aggregate normalized utility. Lines end when the coach is fired; cap hell forces a first-season firing under every policy.

The dynamics-model comparison replaces GPT-5.5 with Claude Opus 5 while holding the v11 prompt, engine, policy models, initial states, seeds, and engine random draws fixed. It crosses 12 policy models, six executed states, and three seeds, for 216 trajectories under each dynamics LLM. The negative-control trajectories are executed but contribute zero to the totals in Table 15.

Table 15: NFL Team Management dynamics-model comparison. Each entry sums mean expected titles over the five scored states, with rank in parentheses. Each mean uses seeds 0–2.

Policy model	GPT-5.5 dynamics	Claude Opus 5 dynamics
GPT-6 Astra	12.580 (1)	10.887 (2)
Claude Fable 5	12.207 (2)	9.420 (5)
Kimi K3	11.440 (3)	9.193 (7)
GPT-5.6 Sol	11.263 (4)	11.540 (1)
Claude Opus 5	11.153 (5)	10.063 (4)
Gemini 3.8 Flash	10.993 (6)	9.203 (6)
GLM-5.3	10.927 (7)	10.427 (3)
Claude Fable 5.1	10.653 (8)	9.033 (9)
Qwen3.8-Max	10.643 (9)	9.087 (8)
GPT-5.6 Luna	10.570 (10)	8.313 (12)
DeepSeek V4 Pro	10.183 (11)	8.547 (11)
Claude Sonnet 5	10.027 (12)	8.627 (10)

Across the 12 models, the two rankings have Spearman correlation $\rho = 0.783$. Claude Opus 5 dynamics gives lower absolute scores for 11 of 12 models. This comparison supports stability of the coarse ordering under this one model substitution; it does not establish that individual ranks or absolute utilities are invariant to the dynamics LLM.

D.5 Historical replay

The replay study uses ten historical coaching tenures, five assigned to development and five assigned to evaluation, with engine seeds 42–46. Historical records initialize the first season and specify major player moves, coordinator changes, scheme changes, and draft trades. Later seasons start from the preceding simulated state rather than the recorded historical state. Competitive posture, spending intentions, and other unobserved decisions are inferred and marked as such. The comparison therefore measures the reconstruction and transition pipeline together.

A diagnostic pooled over all ten tenures informed one change to the final dynamics prompt. The evaluation split is consequently not fully independent of development. Forty-eight of 50 tenure–seed runs complete. Two stop after a salary-cap check fails; statistics for the affected tenures use four seeds.

Table 16 retains the split-level statistics that can be reported without relying on the inconsistent pooled title total. The title statistic is the mean total across seeds, with the standard deviation across seeds. Win error is mean absolute error in regular-season wins.

Table 16: NFL Team Management historical replay by split. Each split contains five coaching tenures. Title correlation is computed across the five tenures; title totals and win error aggregate the completed tenure–seed runs in that split.

Split	Spearman ρ	Simulated titles	Historical titles	Win MAE
Evaluation	0.825	3.76 ± 2.53	4	3.30
Development	0.730	5.21 ± 2.20	7	2.72

Team, coach, and player names are removed from model inputs, although a distinctive roster and action sequence may still reveal a tenure.

E YC-BENCH TRANSITION-REPLACEMENT STUDY

E.1 CONSTRUCTION AND ADAPTER BOUNDARY

The study preserves YC-Bench's startup-management task, commands, task lifecycle, $200,000 initial capital, and funds-based objective. The state includes time, funds, payroll, runway, company status, prestige, employees, clients, tasks, and event history. Employee productivity, client loyalty or adversarial status, and pending-event information are visible to the dynamics model but hidden from the policy model.

The original commands inspect company information, accept and assign tasks, dispatch or cancel work, and advance to the next event. The adapter checks command arguments and preserves the required order of task operations. GPT-5.5 predicts task progress, success or failure, financial changes, payroll, prestige, new tasks, time advances, and terminal events that the Python transition function would otherwise compute. The original Python transition rules do not repair the predicted state in the replacement condition.

E.2 FIXED-ACTION REPLAY

The fixed-action study replays 24 command sequences from the Python simulator under the final dynamics prompt. At day 90, the reported mean relative funds error is 3.91%. Fifteen of 24 sequences match Python funds exactly, and 23 of 24 match the direction of change from the $200,000 initial funds. After pooling sequences by the policy model that generated them, ranks correlate with the reference dataset at Spearman $\rho = 0.762$. Because actions are fixed, this study measures accumulated transition differences without policy adaptation.

E.3 INTERACTIVE EVALUATION AND COMPLETE RESULTS

The interactive comparison crosses eight policy models with seeds 1–3, producing 24 trajectories under LLM dynamics. For each policy model and seed, the Python simulator is also run twice independently. The policy models are GPT-5.5, GPT-5.4-mini, Claude Sonnet 4.6, Claude Opus 4.8, Gemini 3.1 Pro, Gemini 3 Flash, Kimi K2.6, and GLM-5. Policy reasoning uses provider defaults.

For day-90 scoring, the analysis uses the first state at or after day 90, or the closest state if the episode terminates earlier. For each simulator condition, it averages funds over the three seeds for each model and ranks models using the unrounded means. Tables 17 and 18 report the complete comparison.

Table 17: YC-Bench mean funds at day 90, in thousands of dollars, with rank in parentheses. Each mean uses seeds 1–3; ranks use unrounded funds.

Policy model	LLM dynamics	Python run 1	Python run 2
GPT-5.5	902 (1)	264 (2)	255 (3)
Claude Sonnet 4.6	356 (2)	210 (5)	329 (1)
Claude Opus 4.8	353 (3)	275 (1)	314 (2)
Kimi K2.6	182 (4)	211 (4)	236 (4)
Gemini 3.1 Pro	167 (5)	196 (6)	202 (6)
Gemini 3 Flash	166 (6)	192 (7)	174 (7)
GLM-5	166 (7)	211 (3)	216 (5)
GPT-5.4-mini	140 (8)	164 (8)	154 (8)

Table 18: YC-Bench model-ranking agreement. Each ranking is based on eight model-level means, each computed over three seeds.

Comparison	Endpoint	Spearman ρ	p
LLM dynamics vs. Python run 2	Day 90	0.857	0.0065
Python run 1 vs. Python run 2	Day 90	0.738	0.037
LLM dynamics vs. Python run 2	One year	0.571	0.139

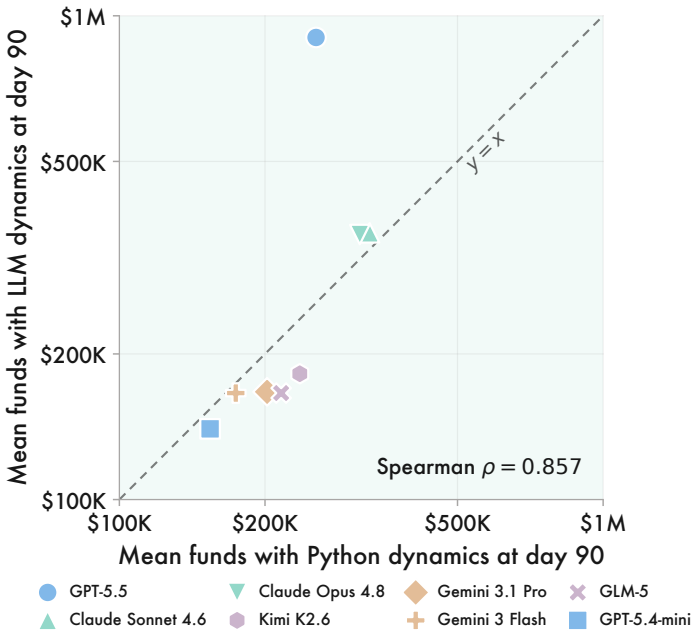

Figure 10: YC-Bench day-90 model-level mean funds under Python run 2 and LLM dynamics. Each point averages seeds 1–3. The rank correlation over the eight models is $\rho = 0.857$.

The correlation is lower at one year. The comparison concerns model ordering; it does not show agreement in absolute funds.