TRADINGARENA: A MARKET-MAKING BENCHMARK FOR RANKING, DIAGNOSING, AND STEERING LLM BEHAVIOR

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language models are increasingly deployed as agents, evaluation should capture both task performance and interactive behavior. Few benchmarks jointly measure performance and behavior under controlled changes to an agent's information, action space, and inference. We introduce **TradingArena**, a multi-agent market-making environment for both performance ranking and controlled behavioral analysis. Agents trade on a shared limit order book using noisy, biased observations of a changing hidden value, with performance measured by mechanically computed profit and loss (PnL). We evaluated 14 models across six configurations and multiple competitive rosters for performance ranking, and further used controlled interventions to diagnose behavioral failure modes. In particular, we quantitatively diagnose over-confidence in GPT-5.5, which trades aggressively under unreliable information and shows a low abstention rate on unanswerable questions. A self-refine intervention reduces this aggressiveness and moves GPT-5.5 from last to second among 14 models by mean PnL in a matched evaluation. These results illustrate how TradingArena can reveal not only differences in model outcomes, but also how behavior changes under intervention.

1 INTRODUCTION

Agent benchmarks evaluate how well large language models complete tasks, but understanding why models perform differently requires examining the decisions behind their scores. In interactive environments, performance reflects how agents interpret observations, choose actions, and respond to feedback. To study these factors, we need environments where the information available to agents and the constraints on their actions can be varied systematically, and the resulting changes in behavior and performance can be observed.

Existing benchmarks evaluate agents in a range of application settings: WebArena (Zhou et al., 2024) and SWE-bench (Jimenez et al., 2024) examine web interaction and software engineering, while Vending-Bench (Backlund & Petersson, 2025), YC-Bench (He et al., 2026), and StockBench (Chen et al., 2026) study business operations and financial trading. These settings provide complementary views of agent performance. Our emphasis is on controlled comparisons within a shared market: varying the information available and the actions permitted, and observing how models' behavior and performance change.

We introduce TradingArena, a multi-agent market-making environment designed for such controlled comparisons (Figure 1). Agents trade on a shared limit order book using noisy, biased private observations of a changing hidden value. We abstract market interaction into controllable components, including observation noise and bias, customer informedness, action constraints, and horizon. These components allow us to vary the demands placed on agents while preserving the consequences of their decisions through trades, inventory, and final settlement. Performance is measured by mechanically computed profit and loss (PnL), while trading records make the agents' behavior directly observable.

Our experiments reveal model-specific behavioral problems and different responses to intervention. In the over-confidence case study, GPT-5.5 combines aggressive trading under unreliable information with a low abstention rate on unanswerable questions. Self-refine reduces its liquidity taking

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.

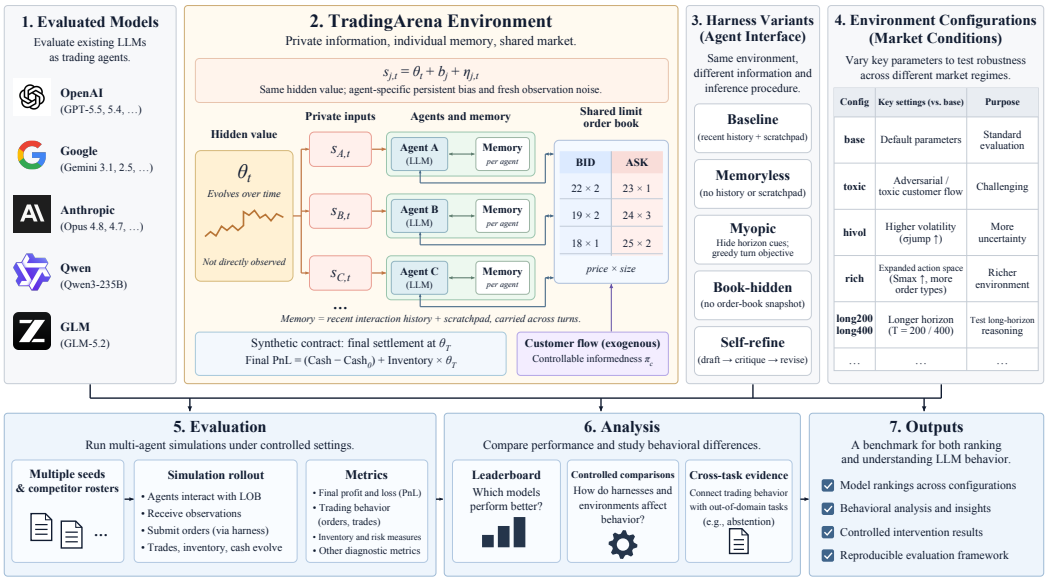

Figure 1: **Overview of TradingArena.** Models interact in a shared market under specified environment configurations and agent interfaces. Games are evaluated across multiple seeds and competitive rosters. Final PnL supports performance comparison, while trading records, controlled interventions, and external evaluations support behavioral analysis.

and order size, turning a negative mean PnL into a positive one and moving it from last to second among 14 models in the matched evaluation. These results connect evaluation with improvement: TradingArena can reveal behavioral weaknesses and test whether changes to an agent's inference procedure mitigate them and improve task performance, without retraining the underlying model.

Contributions:

1. **TradingArena**, a multi-agent market-making benchmark with controllable observations, customer flow, and action constraints, observable trading behavior, and mechanically computed PnL (Section 3).
2. **A comparative evaluation** of 14 frontier models across six market configurations, including comparisons among model-family champions, across the full roster, and within model families (Section 4).
3. **An approach to analyzing differences across models** by combining trading behavior, controlled interventions, and evidence from tasks outside trading (Section 5). Using overconfidence as a case study, we examine how models differ in their trading aggressiveness, responses to self-refine, and abstention behavior, and how these differences relate to performance. Additional analyses examine state tracking and responses to myopic instructions (Appendix D).

2 Related Work

LLM agent evaluation. AgentBench (Liu et al., 2024) evaluates multi-turn decision-making across diverse environments, while WebArena (Zhou et al., 2024) and SWE-bench (Jimenez et al., 2024) assess functional outcomes in web interaction and software engineering. For open-ended responses, LLM-as-judge (Zheng et al., 2023) offers scalable evaluation but can introduce judge biases. TradingArena compares models within a shared market, where agents' actions affect one another's trading opportunities, and examines their responses to controlled interventions.

Table 1: **Primary overall ranking:** the six model-family champions in one finals field (6 makers, 20 seeds). Mean PnL per config and average rank; bold = best in each column.

champion	base	toxic	hivol	rich	long200	long400	avg rank
Gemini 3.1 Pro	**+213**	**+132**	**+287**	**+217**	**+722**	**+1447**	**1.0**
Claude Opus 4.8	+115	+103	+99	+86	+292	+482	2.3
GLM-5.2	+78	+58	+87	+22	+245	+663	3.3
GPT-5.5	+112	−10	+168	−1	+131	+474	3.7
Claude Sonnet 4	+28	−82	+32	+39	+64	+59	4.7
Qwen3-235B	−70	−105	−100	−62	−118	−234	6.0

Economic environments and financial agents. Vending-Bench (Backlund & Petersson, 2025) and YC-Bench (He et al., 2026) evaluate long-horizon business decisions through vending-machine operations and simulated startups. FinMem (Yu et al., 2025) and FinCon (Yu et al., 2024) develop memory-based and multi-agent approaches to financial decision-making. StockBench (Chen et al., 2026) evaluates sequential trading, while ABIDES (Byrd et al., 2019) simulates exchange interactions. TradingArena abstracts market interaction into controllable observations, customer flow, action constraints, and horizon. Rather than reproducing a particular business or market, it uses these components to study changes in model behavior and performance while retaining the economic consequences of trading and settlement.

Self-assessment and behavioral adjustment. Confidence-elicitation studies (Xiong et al., 2024) examine models' assessment of answer reliability, while AbstentionBench (Kirichenko et al., 2025) evaluates abstention when information is insufficient. Self-Refine (Madaan et al., 2023) revises outputs through self-generated feedback; Reflexion (Shinn et al., 2023) uses verbal feedback and memory to guide subsequent attempts. Our case study connects these concerns to trading aggressiveness under unreliable information and responses to intervention.

3 The TradingArena Environment

TradingArena is built to focus on three properties that agent evaluations rarely combine: **(i)** a *dynamic hidden target*—the quantity being estimated evolves stochastically, so an accurate estimate at one turn may become outdated at later turns; **(ii)** *partial and unequal information*—each agent receives private observations with short-term noise and an unknown persistent bias, which cannot be removed simply by averaging observations, and never observes the true value directly; and **(iii)** *intertemporal coupling*— inventory and resting orders persist, so each turn's action constrains the next. A multi-agent, multi-turn market-making game on a limit order book (LOB), settled against a hidden value process, delivers all three (Figure 1). Properties (i) and (ii) hold by construction; (iii) is a design target that the current parameterization may attain only weakly. Appendix D.3 discusses what is needed to assess its planning demands. Formally, TradingArena is an N-player partially observable stochastic game. Section 3.1 defines its components and Section 3.2 maps them onto the agent's state, action, and instruction. Every component serves two roles at once: it mirrors a feature of real market microstructure, and it loads one separable capability that we later examine. Section 3.3 then states what the environment supplies to each half of our study—the design axes that generate the market configurations of Section 4, and the properties that make the controlled comparisons of Section 5 valid measurements rather than coincidences.

3.1 Game definition

To specify the game, we describe the underlying market dynamics, the information and actions available to agents, and how their interactions produce trades and final payoffs. We first define the hidden-value process and private observations, then the action space, transaction rules, and customer flow, and finally the PnL objective.

Hidden value. The contract settles to a latent value θ_t following a jump-diffusion (Merton, 1976):

$$\begin{gathered}\theta_t = \theta_{t-1} + \kappa(\mu - \theta_{t-1}) + w_t + \mathbf{1}[t \in J]\,\xi_t, \\ \theta_1 \sim \mathcal{N}(\mu, (\sigma_\theta \mu)^2), \quad w_t \sim \mathcal{N}(0, (\sigma_w \mu)^2), \quad \xi_t \sim \mathcal{N}(0, (\sigma_{\text{jump}} \mu)^2).\end{gathered} \tag{1}$$

The process combines mean reversion toward μ at rate κ, zero-mean Gaussian fluctuations w_t, and jumps ξ_t on a fixed set of turns J. w_t and ξ_t stand for the two ways a real asset's worth moves. The diffusion term w_t represents ordinary price fluctuations associated with routine flow and minor news. The jump ξ_t is a discrete repricing on a date the whole market has in its calendar—an earnings release, a central-bank decision, a macro data print. Like those events it is flagged in advance: one turn ahead, every agent receives the same jump warning, giving the sign of ξ_t but never its magnitude. Each σ above is a fraction of μ rather than an absolute price, so only these ratios matter.

Observations. Agent j never sees θ_t itself. Each turn it receives *a single private, noisy read of* s,

$$s_{j,t} = \theta_t + b_j + \eta_{j,t}, \qquad b_j \sim \mathcal{N}(0, (\sigma_b \mu)^2), \quad \eta_{j,t} \sim \mathcal{N}(0, (\sigma_\eta \mu)^2), \tag{2}$$

together with *the market state*. Here b_j is the per-agent bias, drawn once per game and fixed for its duration. $\eta_{j,t}$ is fresh per-turn noise, the chance error in any single reading. It sets how informed agent j is: a small $|b_j|$ is a well-informed trader, a large one a trader working from a skewed view. b_j and $\eta_{j,t}$ are not equally removable, and that asymmetry is what makes filtering non-trivial. Averaging an agent's own observations reduces $\eta_{j,t}$, whereas the persistent bias b_j survives self-averaging and must be inferred by cross-checking other agents' order flow against its own observations. All of w_t, ξ_t, b_j, and $\eta_{j,t}$ are mutually independent.

The market state exposed to agent j has two parts: a snapshot of the agent's cash, inventory $I_{j,t}$ (its net position), open orders, and the full order book; and an ordered *event stream* of fills, accepts, and cancels since the agent last acted. Counterparties are anonymized, while order IDs remain stable across turns.

Actions. Each turn, agent j may submit at most one bid and one ask, together with cancellations of any of its own resting orders. Each new order is specified by $(\text{side}, \text{price}, \text{size}, \text{type})$, where $\text{size} \in \{1, \ldots, S_{\max}\}$. A typical market-making action places both, but an agent may quote only one side or submit no new orders at all.

An accepted limit order that crosses an existing quote at execution time trades at the resting order's price and *takes* liquidity. Any unfilled remainder rests on the book, where it can later trade as a *maker*. Thus, make/take behavior is determined mechanically by the matching engine, allowing aggressiveness to be measured directly from execution logs. We additionally support explicit order types, including `post-only`, `IOC`, `FOK`, and `market` (Appendix A.1).

Each agent maintains an inventory $I_{j,t}$, defined as its net position (units bought minus units sold) at turn t. The action space is bounded by a maximum order size $S_{\max}$ and an inventory limit $I_{\max}$. The reference setting uses $S_{\max} = 1$ and limit orders only. Orders that would exceed the inventory bound $\pm I_{\max}$ are rejected. Resting orders remain on the book until filled or canceled.

Transaction Rules. We call the position a model occupies in a game its *seat*. At each turn, all seats act on the book state from the end of the previous turn. Agent submissions are then sent to the matching engine in a uniformly random order determined by the game seed, followed by customer submissions. This turn structure mimics the observe–act latency of real-world trading: agents make decisions from an observed book state, while the market may change before their orders reach the engine. The random arrival order among agents prevents any seat from having a systematic first-mover advantage, making seats exchangeable in distribution, a property used in Section 3.3 . The engine follows price–time priority: orders at better prices execute first, and among resting orders at the same price, earlier arrivals have priority.

Customers. Exogenous flow comes from a scripted minority of *customers*: liquidity-demanding counterparties that hit the makers' quotes so the market trades at all. Each customer c carries an *informedness* $\pi_c \in [\frac{1}{2}, 1]$, the probability that its next order trades in the direction favored by the hidden value—buying when the asset is undervalued and selling when it is overvalued. At $\pi_c = \frac{1}{2}$, the flow is uninformed. As $\pi_c \to 1$, the customer increasingly trades in the direction favored by the hidden value, creating adverse selection for market makers (Glosten & Milgrom, 1985).

Objective. A game runs for a fixed horizon of T turns. At expiry the contract settles to θ_T and each agent's score is

$$\text{Final PnL} = (\text{Cash} - \text{Cash}_0) + \text{Inventory} \times \theta_T, \tag{3}$$

where Cash_0 is the agent's starting cash. Cash may run negative, so agents face no funding constraint; inventory may be long or short, subject to the position limit. Whatever inventory an agent still holds at expiry is marked against the true θ_T: a loss cannot be avoided by never closing the position. The score is a single scalar, settled mechanically from the executed trajectory, with no judge and no rubric.

3.2 From environment to agent: state, action, instruction

The harness is deliberately simple: a ReAct-style loop (Yao et al., 2022) with a bounded history window and a free-form scratchpad (Nye et al., 2021); turn I/O is structured JSON. At each turn, the model receives the raw noisy observation $s_{j,t}$ (under the field `model_estimate`). A strict design constraint is that the harness performs no filtering and does not supply the true hidden value θ_t to the model. The model must therefore integrate and denoise observations across turns on its own, so differences in how models handle unreliable information reflect model behavior rather than additional processing by the harness.

Standardized Agent Interface. All models share a model-agnostic task template, which specifies the market-making role, JSON action schema, position limit, and objective (maximize Final PnL). The prompt explains that `model_estimate` contains noise and persistent systematic bias, but provides no trading strategy: it specifies no target spread, inventory-skewing rule, or make/take strategy. Prompts are never tuned for individual models.

All ablations in Section 5 and Appendix D are explicit modifications of this common baseline. *Memoryless* removes retained dialogue and scratchpad while keeping the current state and previous-turn events; *myopic* hides total and remaining horizon and adds a greedy instruction; *book-hidden* replaces the order-book snapshot with public order-flow events; and *self-refine* adds a critique-and-revision call whose added instruction mentions neither trading nor the evaluation metric. Interface details and prompt-reproduction requirements are documented in Appendix A.2.

Table 5 in Appendix A.2 maps each environment variable in Section 3.1 to its role in the agent loop, its representation in the harness, the capability it tests, and the corresponding measurement.

3.3 Two roles: leaderboard and instrument

TradingArena supports two complementary evaluation modes: varying the environment to rank models across market conditions, and using the environment as an instrument to examine specific capabilities through interventions on the agent.

As a leaderboard: Model ranking across market conditions. TradingArena supports model ranking across controlled variations of the market environment. We vary four environment parameters: the jump scale σ_{jump} (Eq. 1), which controls the magnitude of sudden changes in the hidden value; the customer informedness vector $\{\pi_c\}$, which controls the degree of adverse selection in exogenous trading flow; the per-order cap $S_{\max}$ and enabled order types, which constrain the agent's action space; and the horizon T (Eq. 3), which controls the temporal length of the game. Varying one factor at a time from a common reference configuration produces markets that differ along a single, well-defined dimension. Section 4 uses these configurations to measure both overall performance and how model performance changes across market conditions.

As an instrument: controlled behavioral analysis. To analyze model behavior, we change an agent's information or inference procedure while keeping the roster and environment settings fixed. We compare trading behavior and PnL over matched seeds, without assuming that the resulting market trajectories remain identical. Section 5 presents an over-confidence case study, with additional analyses in Appendix D. Appendix A details the experimental protocols and metric definitions.

4 TradingArena as a Leaderboard

Because a model's performance can depend on the other models in the same game, our *primary* overall ranking compares model-family champions. We complement this with an all-model comparison and within-family generational comparisons.

We report mean PnL over 20 seed-paired runs per configuration. Settings differ across competitive rosters; Appendix A.3 provides the full parameter values and customer specifications for each field.

4.1 Roster and configurations

We evaluate 14 frontier models across six model families from five providers: GPT-5.5, GPT-5.4, GPT-5.2 (Singh et al., 2025), and GPT-4.1 (Achiam et al., 2023) (OpenAI); Claude Opus 4.8, Opus 4.7, Opus 4.6, and Sonnet 4[1] (Anthropic); Gemini 3.1 Pro, Gemini 3.5 Flash, Gemini 2.5 Pro, and Gemini 2.5 Flash[2] (Google); Qwen3-235B-A22B (Yang et al., 2025) (Alibaba); and GLM-5.2 (Zeng et al., 2026) (Zhipu AI). We treat Claude Opus and Claude Sonnet as separate families in our comparisons. We evaluate models across six market configurations built from the environment variations introduced in Section 3.3: jump scale, customer informedness, action space, and horizon. Exact parameter settings for all six are listed in Table 6; their design intent is:

- **base** — the reference configuration, using the default hidden-value process, customer flow, action space, and 100-turn horizon. All other configurations modify one of these components while keeping the rest fixed.
- **toxic (Customers)** — increases customer informedness $\{\pi_c\}$, testing how well agents recognize and adapt to unreliable estimates under stronger adverse selection.
- **hivol (Hidden value)** — increases the jump scale σ_{jump} of ξ_t , testing how quickly agents update their estimate when the hidden value changes abruptly, placing greater demand on *state tracking*.
- **rich (Action)** — expands the action space by increasing the per-order size cap $S_{\max}$ and enabling all supported order types, testing whether agents use the added flexibility without acting too aggressively, thereby exposing *over-confidence* and over-action.
- **long200 / long400 (Horizon)** — extends the horizon T from 100 to 200 or 400 turns, testing whether agents account for future consequences over longer interactions, placing greater demand on *planning*.

To examine how individual environment parameters affect model performance and trading behavior, we conduct additional experiments varying jump magnitude, persistent observation bias, observation noise, ordinary diffusion, inventory capacity, and mean reversion one at a time. These comparisons help characterize the effects of each parameter and contextualize the default settings. Appendix B presents the parameter choices and detailed results.

4.2 Overall ranking: family champions

Our *primary* overall ranking is a finals among family champions: we take each model family's *champion*—its best version, chosen from the within-family ladders (Section 4.3)—and run the six champions (GPT, Claude Opus, Claude Sonnet, Gemini, Qwen, GLM) in a single field (6 makers, 20 seeds). With every seat held by a family's best version, there is little weak opposition left to *farm*. Table 1 reports the finals across all six configs.

Gemini 3.1 Pro wins every configuration (avg rank 1.0), the long-horizon configs amplifying its lead (long400 $+1447$). **GLM-5.2 places third**, ahead of GPT-5.5, and is positive in every config; **GPT-5.5** is dragged down by `toxic` (-10) and `rich` (-1)—the over-trading signature we dissect in Section 5—while Qwen trails throughout.

4.3 Within-family ladders

To compare versions *within* a family, we run family-internal fields (one family per game). Table 2 summarizes three families across selected configurations (mean PnL; bold = family winner).

Gemini: Gemini 3.1 Pro wins every configuration by large margins, but ordering tracks *tier* over recency—the later Gemini 3.5 Flash places only second, behind the 3.1 *Pro*; **GPT** is non-monotone

[1] `https://www.anthropic.com/`

[2] `https://deepmind.google/models/gemini/`

Table 2: Family-internal generational ladders: mean PnL per model version and configuration (20 seeds). Bold marks the highest mean PnL within each family and configuration.

version	base	toxic	hivol	rich	long400
Gemini					
Gemini 3.1 Pro	**+557**	**+336**	**+460**	**+443**	**+3320**
Gemini 3.5 Flash	+140	−89	+172	+168	+316
Gemini 2.5 Pro	+43	−35	+56	−97	−179
Gemini 2.5 Flash	−4	−215	−28	−146	−227
GPT					
GPT-5.5	**+380**	**+124**	**+408**	+97	**+2165**
GPT-5.4	+90	+41	+71	+131	+371
GPT-5.2	+16	−121	+56	+111	−124
GPT-4.1	+200	−32	+166	**+155**	+306
Opus					
Opus 4.8	**+51**	**−30**	**+64**	+35	**+152**
Opus 4.7	+50	−46	+36	**+38**	+112
Opus 4.6	+8	−55	+35	+4	+100

Table 3: All-in-one mixed field: mean Final PnL, 14 models × 6 configs, 20 seeds each. Sorted by average rank; bold = best in each column.

model	base	toxic	hivol	rich	long200	long400	avg rank
Gemini 3.1 Pro	**+244**	**+204**	**+228**	+35	**+619**	**+1494**	**1.7**
Opus 4.6	+171	+154	+212	+53	+429	+1234	2.8
GLM-5.2	+199	+131	+140	**+77**	+406	+1362	2.8
Opus 4.7	+179	+141	+184	+73	+345	+1181	3.3
Opus 4.8	+142	+129	+162	+64	+375	+1084	4.3
GPT-5.5	+67	+17	+127	−158	+106	+79	8.0
Gemini 2.5 Pro	+59	+37	+74	−47	+88	+59	8.3
GPT-4.1	+29	−6	+38	−6	+18	+144	8.8
Gemini 3.5 Flash	+14	+12	+68	+34	+83	−0	9.0
GPT-5.4	+21	+43	+86	−12	+12	−92	9.5
Sonnet 4	+23	−23	+17	+11	+38	+121	9.5
GPT-5.2	+24	+12	+32	−10	−23	−125	10.7
Qwen3-235B	−42	−86	−86	−67	−85	−44	12.8
Gemini 2.5 Flash	−76	−98	−54	−19	−148	−421	13.3

(the *oldest*, GPT-4.1, wins `rich`—the over-trading of the newest model again); **Opus** 4.8 achieves the highest mean PnL in four of the five reported configurations, while Opus 4.7 leads in `rich`.

4.4 ALL-IN-ONE FIELD

As a complementary view, all 14 models compete in one mixed field; note this regime *rewards farming*, so it is secondary to the champion finals above. Table 3 reports mean PnL per model × config and the average rank.

Three observations. (i) **Gemini 3.1 Pro leads overall** (avg rank 1.7), topping five of six configs, with long-horizon configs amplifying the leader (long400 $+1494$). (ii) **GLM-5.2 is a standout**—tied for second on average rank and positive in every config—and the Anthropic Opus family fills out the upper tier (Opus 4.6/4.7/4.8). (iii) **GPT-5.5 drops to mid-pack**, collapsing in `rich` (-158): opening the action space (large size + all order types) surfaces an over-trading signature we dissect in Section 5. Appendix C.1 examines how participation, liquidity taking, and inventory differ across models. Appendices C.2 and C.3 connect these aggregate measures to specific observations, orders, and fills in two archived games.

5 TRADINGARENA AS AN INSTRUMENT

A PnL ranking says who performs better, but does not explain the behaviors behind the differences. We use TradingArena as an instrument to analyze behavioral problems and performance differences across models. Our approach starts with defining a capability limitation in task-general terms, then

identifying observable behaviors that may reflect it in the trading environment. We then apply controlled interventions and examine whether they change those behaviors and improve PnL. Finally, we compare the findings with performance on an external task that may reflect the same limitation. Agreement across these perspectives strengthens the interpretation of the observed behavior.

We illustrate this approach through **over-confidence**, examining how different models act on unreliable information and respond to intervention. Trading behavior, responses to self-refine (Madaan et al., 2023), accompanying PnL changes, and performance on unanswerable questions provide consistent evidence of over-confidence as a failure mode in **GPT-5.5**, while comparisons with other models help distinguish aggressive trading from other sources of performance differences. Appendix D explores state tracking and planning.

5.1 Observable behavior: trading aggressiveness

We use *over-confidence* to describe over-trusting unreliable information and acting too aggressively on it. In the arena, agents must trade using noisy, biased observations of the hidden value. We examine *taking liquidity* by crossing existing quotes. Taking liquidity means accepting the opposite side's quoted price—for example, buying at the ask rather than posting a bid and waiting—thus paying the bid–ask spread for immediate execution. This behavior indicates trading aggressiveness. To assess whether that aggressiveness reflects over-confidence in unreliable observations, we examine how it changes under self-refine and whether those changes accompany improved PnL. For a concrete illustration of why orders and actual fills must be distinguished, Appendix C.2 follows three models through the same value jump; these baseline examples are separate from the intervention experiments below. We examine the 14-model field under the `rich` configuration of Section 4, using 10 seeds. Our primary behavioral indicator is **taker%**: the fraction of an agent's fills in which it was the crossing taker. Table 4 reports this indicator alongside intervention effects for the nine tested models. Baseline taker shares differ substantially: GPT-5.4 is at 48.9%, whereas Gemini 3.1 Pro and Opus 4.8 are at 15.6% and 15.2%, respectively. GPT-5.5 has the highest taker share among all 14 baseline models (72.6%). These differences motivate examining how reflection changes each model's behavior and performance.

5.2 Controlled intervention: self-refine

Self-refine (Madaan et al., 2023; Shinn et al., 2023) uses a draft call followed by one combined critique-and-revision call each turn. The added instruction provides no new market facts or trading strategy and mentions neither trading nor the evaluation metric. It does explicitly ask the model to check for over-reaction to its most recent input (Appendix A.2). We apply self-refine separately to nine models in the 14-model field over the same 10 seeds. For each tested model, we compare its baseline seat with the same seat in a separate run using the same seed and roster, with only that seat switched to self-refine. All other seats retain their baseline inference procedure. The environment settings are fixed, although the resulting market trajectories can differ as agents respond to the changed actions. Self-refine can affect several aspects of reasoning; its behavioral and PnL effects are therefore interpreted jointly.

Behavioral changes. Alongside changes in taker%, we examine changes in mean quote size as a complementary indicator of how self-refine affects trading aggressiveness. For GPT-5.5, self-refine reduces taker% by 20.5 percentage points and nearly halves mean quote size ($2.45 \rightarrow 1.26$; Table 4). These two changes consistently indicate more restrained trading after reflection. Across the nine tested models, higher baseline taker% is associated with a larger reduction under self-refine (Spearman rank correlation $\rho(\text{baseline taker\%}, \Delta\text{taker}) = -0.93$), where Δtaker = self-refine − baseline, as in Table 4. GPT-5.4 shows the largest taker% reduction (-32.7 percentage points), followed by GPT-5.5 (-20.5 percentage points).

PnL changes. The same intervention improves GPT-5.5's mean PnL by 220 (Table 4). Its lower taker share and smaller orders therefore accompany a substantial performance improvement. Together, these outcomes support the interpretation that its baseline trading was overly aggressive, although they do not isolate reduced aggressiveness as the sole cause of the PnL gain.

Table 4: Trading behavior, self-refine effects, and external abstention results. Arena results cover nine tested models in the 14-model `rich` field (10 paired seeds; seed selection in Appendix A.4). Δ = self-refine − baseline; pp denotes percentage points. Abstention rates are from the LLM detector on AbstentionBench for all nine models. External evaluation details are in Appendix D.1. Models are ordered by baseline taker%. Bold marks the highest baseline taker%, largest behavioral reductions, and largest PnL improvement.

	Taker share		Mean quote size		PnL	External
model	baseline (%)	Δ (pp)	baseline	Δ	ΔPnL	Abstention (%)
GPT-5.5	**72.6**	−20.5	2.45	**−1.19**	**+220**	42.3
GPT-5.4	48.9	**−32.7**	1.05	+0.03	−3	44.3
Gemini 2.5 Pro	46.5	−2.2	2.20	−0.29	+59	45.5
Gemini 3.5 Flash	28.1	−11.3	2.18	+0.26	+10	42.7
Opus 4.7	19.3	−9.3	1.64	−0.38	+26	47.8
Qwen3-235B	17.0	−0.2	2.28	−0.67	+72	47.7
GLM-5.2	16.4	+0.4	1.78	−0.22	+27	46.1
Gemini 3.1 Pro	15.6	+2.1	3.56	+0.30	+125	44.5
Opus 4.8	15.2	+5.2	1.59	−0.18	−8	52.4

The relationship differs across models. GPT-5.4 reduces its taker share most but gains no PnL (ΔPnL = −3). Gemini 3.1 Pro gains +125 despite a low baseline taker share (15.6%) and increases in both taker share and quote size; Qwen3-235B gains +72 with almost no change in taker share. These cases show that self-refine can improve PnL without reducing the same behaviors, and that less active trading is not universally more profitable. For GPT-5.5, reduced aggressiveness accompanies improved performance; for GPT-5.4, the same directional change in taker share is insufficient; for Gemini 3.1 Pro and Qwen3-235B, gains occur without reduced taker share. The arena thus helps distinguish different responses to reflection, while the mechanisms behind each model's improvement require further investigation.

5.3 External validation: abstention on unanswerable questions

We use **AbstentionBench** (Kirichenko et al., 2025), a benchmark of whether models abstain on unanswerable questions, to examine over-confidence outside trading. Answering despite insufficient information is consistent with the tendency to act on unreliable evidence that we study in the arena. Among the nine models evaluated on AbstentionBench, GPT-5.5 has the lowest abstention rate (42.3%). This complements the arena evidence: it has the highest baseline taker share (72.6%), which falls by 20.5 percentage points under self-refine, and its mean quote size shows the largest reduction among the nine intervention models (−1.19), alongside a PnL improvement of 220. Opus 4.8, in contrast, has the highest abstention rate (52.4%) and the lowest baseline taker share (15.2%); self-refine does not reduce its taker share or improve its PnL, and its mean quote size decreases only slightly (−0.18) (Table 4). Across these nine models, arena taker% and abstention rate have a Spearman rank correlation of $\rho = -0.70$: models that trade more aggressively also tend to abstain less, providing external support for our interpretation of their arena behavior.

6 Conclusion

We introduced TradingArena, a controllable multi-agent market-making environment for comparing model performance and analyzing behavior. Across 14 models and six configurations, our experiments reveal differences in both trading performance and responses to intervention. The over-confidence case study connects aggressive trading under unreliable information with a tendency to answer unanswerable questions. For GPT-5.5, the combination of aggressive trading, low abstention on unanswerable questions, and improved PnL under self-refine provides consistent evidence of over-confidence as a behavioral problem.

AI USE STATEMENT

We used generative AI tools for language polishing and manuscript editing, and to assist with coding, experimental design discussions, and result analysis. The authors take responsibility for the final content of the paper.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Axel Backlund and Lukas Petersson. Vending-bench: A benchmark for long-term coherence of autonomous agents. *arXiv preprint arXiv:2502.15840*, 2025.

David Byrd, Maria Hybinette, and Tucker Hybinette Balch. Abides: Towards high-fidelity market simulation for ai research. *arXiv preprint arXiv:1904.12066*, 2019.

Yanxu Chen, Zijun Yao, Yantao Liu, Amy Xin, Jin Ye, Jianing Yu, Lei Hou, and Juanzi Li. Stockbench: Can llm agents trade stocks profitably in real-world markets?, 2026. URL https://arxiv.org/abs/2510.02209.

Lawrence R Glosten and Paul R Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics*, 14(1):71–100, 1985.

Muyu He, Vincent Tu, Adit Jain, Anand Kumar, Sachin Patro, Soumyadeep Bakshi, and Nazneen Rajani. YC-bench: Benchmarking AI agents for long-term planning and consistent execution. In *ICML 2026 Workshop on Combining Theory and Benchmarks: Towards A Virtuous Cycle to Understand and Guarantee Foundation Model Performance*, 2026. URL https://openreview.net/forum?id=wHPdBWy4xd.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Polina Kirichenko, Mark Ibrahim, Kamalika Chaudhuri, and Samuel J. Bell. Abstention-bench: Reasoning llms fail on unanswerable questions. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/fb122bfc3f0127a94ded048b5b03496f-Paper-Datasets_and_Benchmarks_Track.pdf.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf.

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

Robert C Merton. Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144, 1976.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. 2021.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 23650–23678, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6733cf15e10e2cd1d59af033c3bb8507-Paper-Conference.pdf.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Yangyang Yu, Zhiyuan Yao, Haohang Li, Zhiyang Deng, Yuechen Jiang, Yupeng Cao, Zhi Chen, Jordan W Suchow, Zhenyu Cui, Rong Liu, et al. Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making. *Advances in Neural Information Processing Systems*, 37:137010–137045, 2024.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Jordan W Suchow, Denghui Zhang, and Khaldoun Khashanah. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *IEEE Transactions on Big Data*, 2025.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

A Environment and Experimental Details

This appendix provides details of the environment rules, agent interface, experimental configurations and rosters, and metric aggregation procedures used in our evaluations.

A.1 Environment rules and accounting

The hidden-value and observation processes are defined in Eqs. 1 and 2. A persistent bias is sampled once for each agent and game; observation noise is sampled each turn. The public jump warning reveals the next jump's direction, not its magnitude. Hidden values, realized biases, and terminal settlement values are not revealed to agents during play.

Agents act on the previous end-of-turn book. Their submissions are processed in a seed-dependent random order, followed by the scripted customers. Consequently, the book can change between an agent's observation and execution. This represents a simplified form of observation-to-execution delay in real-world trading, allowing us to examine how agents manage execution risk when acting on potentially outdated market information. An intended crossing order need not actually take liquidity. The matching engine determines the maker/taker role of each fill and executes at the resting order's price, subject to price–time priority. Orders that would exceed the inventory limit are rejected in full; unfilled resting orders persist until canceled or filled. Passing does not remove them.

Customer direction. Each customer submits a one-contract market order. With both book sides available, its reference price is the best-bid/best-ask midpoint; with only one side, it uses that side's best quote. The value-favored direction is buy when the hidden value exceeds this reference and sell otherwise. The customer chooses that direction with probability π_c and the opposite direction with probability $1 - \pi_c$, so $\pi_c = 0.5$ is directionally uninformed. If the chosen side has no available liquidity, the order produces no fill and is not resampled.

Order types. A `limit` order sets the worst acceptable execution price: it can execute immediately if it crosses, and otherwise rests on the book. A `post-only` order is rejected if it would cross immediately. An `IOC` order executes the available quantity immediately and cancels the remainder; a `FOK` order requires immediate execution of the full quantity or is canceled. A `market` order has no price limit and seeks immediate execution against available quotes. The reference configuration allows only limit orders; `rich` enables all five types. The larger action space does not remove inventory constraints.

Scoring. Final PnL is cash change plus terminal inventory valued at θ_T (Eq. 3). With zero initial cash it is $\text{Cash}_{j,T} + I_{j,T}\theta_T$. Given the executed trades and terminal settlement value, final PnL is fully determined.

A.2 Agent interface and prompt documentation

The baseline interface combines the current private observation (`model_estimate`), cash and inventory, open orders, the order-book snapshot, recent events, a bounded interaction history, and a scratchpad. The baseline prompt explains that the private observation is noisy and contains persistent systematic bias, without prescribing a trading strategy. The current-turn JSON contains the following fields:

- `turn`: the current turn index.
- `cash`: the agent's current cash balance.
- `inventory`: its signed net position; positive values denote a long position and negative values a short position.
- `events`: time-ordered trades and order-status updates since the agent's previous turn, including order acceptance, expiration, rejection, and cancellation outcomes.
- `your_open_orders`: the agent's currently resting orders, each described by its identifier, side, price, and remaining quantity.
- `model_estimate`: the environment-provided private observation $s_{j,t}$, containing noise and persistent bias, not an estimate generated by the LLM agent.

Book mode additionally includes `order_book`, which lists resting buy and sell orders with their stable identifiers, prices, quantities, and an indicator of whether each order belongs to the agent. Flow mode instead exposes public order updates with stable identifiers while retaining the agent's own open orders. Before a scheduled jump, `jump_warning` indicates the upcoming jump's direction, but not its magnitude. Appendix E.1 provides the full baseline prompt, including the nested order fields and event types.

The agent responds with one JSON object containing the following fields:

- `thoughts`: the agent's textual explanation of its assessment of value, market state, and trading strategy.

- `commands`: a list of command strings specifying the actions to submit to the exchange. In the base configuration, `quote` submits bid and/or ask orders, `cancel` requests cancellation of orders by identifier, and `pass` takes no trading action. The `rich` configuration additionally permits the order types described in Appendix A.1.
- `scratchpad` (optional): text that replaces the agent's persistent notes, which are included in subsequent turns.
- `scratchpad_append` (optional): text appended to those existing notes without replacing them.

The harness parses the command strings into structured actions. Parsed side sizes are clamped to their allowed range, and a skipped quote side uses -1. For example, the following parsed quote command is taken from the first turn of Gemini 3.1 Pro in Case 2 (Appendix C.3):

```
{"action": "quote", "bid": 99.0, "ask": 101.0,
 "bid_size": 1, "ask_size": 1,
 "bid_type": "limit", "ask_type": "limit"}
```

This example shows a parsed action, not the full model response: the model emits command strings inside JSON, which the harness parses into records such as this one. Observing a submitted quote alone does not establish whether it was accepted or filled.

History and scratchpad. The harness retains the last 20 user/assistant turn pairs and halves the retained history after a context-overflow exception. The scratchpad is persistent text reinjected into the system prompt, with no explicit code-level length cap; it remains subject to the model's context limit. A new agent instance is created for each game, so state is not carried between games. No external tools are exposed to the models.

Memoryless and myopic instructions. Memoryless retains only the current system and user messages, does not update the scratchpad, and adds the following instruction (line breaks adjusted):

```
You have NO memory of previous turns -- each turn you see
only the current game state and all information revealed
so far. There is no scratchpad and no conversation history.
```

The wording does not supply the full past event stream: the actual current payload retains the state and previous-turn events described above. Myopic removes total horizon from the contract description and changes the state heading from `Turn t/T` to `Turn t`. Turn index, history, scratchpad, observations, events, and jump warnings remain available. It replaces the objective instruction with:

```
You are completely impatient: each turn, maximize this
Final PnL as evaluated RIGHT NOW -- (Cash - Starting Cash)
+ Inventory * (your current best estimate of the settlement
value). Use everything you have learned so far to form
that estimate, but put zero weight on how today's action
affects later turns: do not plan ahead, do not anticipate
future price moves, and do not build or unwind positions
for the future.
```

Self-refine instruction and execution. The self-refine implementation first obtains and parses a draft, then appends that draft as an assistant message and the following user instruction to the same base dialogue (line breaks adjusted for display):

```
Before this is submitted, act as a critic of your own
proposed response above.
1. CRITIQUE it: point out any concrete mistakes, internal
inconsistencies, unjustified jumps, or places where you may
be over-reacting to the most recent input instead of
weighing the full situation you were given.
2. Then produce an IMPROVED final response that addresses
your critique. If the original was already best, you may
keep it.
```

Table 5: **Environment → agent mapping.** Each variable of the game definition (Section 3.1), its role in the agent loop, where it surfaces in the harness, the capability it loads, and where that capability is measured. The last column links this section to the studies that follow.

Environment	Agent-side role	Harness surface	Capability loaded	Section
θ_t (Eq. 1)	estimation target	absent from prompt	state estimation	§D.2
$s_{j,t}$ (Eq. 2)	observation	`model_estimate`, raw	filtering an unreliable input	§5
b_j (persistent bias)	unknown systematic error	never shown; infer from flow	calibration; self-distrust	§5
$\eta_{j,t}$ (fresh noise)	unknown transient error	never shown	cross-turn evidence integration	§D.2
book + event stream	observable market state	snapshot + events	state tracking / reconstruction	§D.2
bid/ask quote; order types	action space	JSON action schema	acting vs. waiting	§5
inventory ($\pm I_{\max}$), resting orders	persistent controlled state	carried across turns	planning under constraint	§D.3
jump warning (sign of ξ_t)	public partial signal	JSON field	acting under known-unknowns	§D.3
customers (π_c)	exogenous / adversarial flow	scripted, not prompted	adverse-selection risk	§5
Final PnL (Eq. 3)	objective	terminal scalar	end-to-end coherence	§4

```
Respond with ONLY a raw JSON object using the SAME schema
as before (thoughts, optional scratchpad/scratchpad_append,
commands).
```

This yields two calls on a normal successful treated turn, rather than separate draft, critique, and revision calls. The added instruction supplies no new market observation or trading prescription, but explicitly addresses recency over-reaction. The surrounding context still includes the trading task, observation, history, scratchpad, and draft. The same intervention procedure is used across the nine models.

Generation settings and failures. The standard generation settings specify a maximum of 32,768 output tokens and request thought output, without setting explicit temperature or nucleus-sampling values. The GLM interface sets temperature 1, nucleus-sampling probability 1, the same output limit, and minimal reasoning effort. The main call allows one retry and a 480-second timeout; exhausted API or parsing failures yield a pass and an error record. An invalid or failed refinement falls back to the parsed draft.

Appendix E provides the full baseline prompt for the planning configuration.

A.3 Configurations and experimental fields

Table 6 specifies the six configurations of Section 4. All use $\mu = 100$, $\sigma_\theta = 0.20$, $\sigma_w = 0.02$, $\sigma_b = 0.05$, $\sigma_\eta = 0.20$, $I_{\max} = 5$, zero initial cash, and zero fees. Sigma values in the paper are fractions of μ; for example, $\sigma_\eta = 0.20$ corresponds to an observation-noise standard deviation of 20 when $\mu = 100$. Customer informedness π_c is distinct from the observation-bias scale σ_b. Unless otherwise noted, jumps are scheduled every 25 turns, and a field with N market makers includes $\max(1, \lfloor N/3 \rfloor)$ customers. Cash may go negative; agents face no funding constraint.

The six-model champion experiments use $\kappa = 0.05$ and 20 seeds. The 14-model experiments use $\kappa = 0$ and 20 seeds.

The champion roster is GPT-5.5, Opus 4.8, Gemini 3.1 Pro, Sonnet 4, Qwen3-235B, and GLM-5.2. The 14-model roster is listed in Section 4. The GPT-family comparisons use four-seat games (GPT-5.5, GPT-5.4, GPT-5.2, GPT-4.1), and the Opus-family comparisons use three-seat games (Opus 4.8, Opus 4.7, Opus 4.6), with 20 seeds and $\kappa = 0.05$. Both use one customer, with informedness 0.5 except in `toxic`, where it is 0.9. The Gemini-family field contains Gemini 3.1 Pro, Gemini 3.5 Flash, Gemini 2.5 Pro, and Gemini 2.5 Flash.

Table 6: Market configurations and customer informedness. Values not changed here retain their field-specific defaults. The `rich` configuration changes both order size and supported types.

Configuration	T	σ_{jump}	$S_{\max}$	Order types
base	100	0.15	1	limit
toxic	100	0.15	1	limit
hivol	100	0.30	1	limit
rich	100	0.15	10	all five
long200	200	0.15	1	limit
long400	400	0.15	1	limit

Field	Non-toxic customers	Toxic customers
Family champions (6 models)	$[0.5, 0.5]$	$[0.9, 0.9]$
All-model field (14 models)	$[0.5, 0.5, 0.8, 0.8]$	$[0.8, 0.9, 0.8, 0.9]$
Gemini family (4 models)	$[0.5]$	$[0.9]$
GPT family (4 models)	$[0.5]$	$[0.9]$
Opus family (3 models)	$[0.5]$	$[0.9]$

A.4 Seeds, metrics, and comparisons

Leaderboard means use 20 seeds per configuration. Section 5 uses the first 10 seeds of the 14-model `rich` field, because each intervention requires another full-field run. Baseline quantities are recomputed on that subset.

For model j and matched seed set $\mathcal{S}$, intervention effects use

$$\Delta\text{PnL}_j = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \left(\text{PnL}_{j,s}^{\text{intervention}} - \text{PnL}_{j,s}^{\text{baseline}} \right).$$

Only the focal seat is switched to self-refine. The other seats retain their baseline inference procedure, but their actions and the resulting market trajectory can change in response. Matching seeds and settings does not hold those endogenous responses fixed.

Within one configuration, rankings order models by mean PnL. An average rank across configurations is a different statistic, as is the average of per-game ranks. The reported GPT-5.5 change from last to second uses rankings by mean PnL in the two matched 10-seed fields, not insertion of its new score into an unchanged baseline leaderboard.

Taker share is computed within each game by counting fill records involving the model and dividing the number in which it was the taker by that count. The over-confidence analysis requires at least five fill records and averages eligible game-level percentages. It does not pool contract volume across games; no-fill games are omitted, not assigned zero.

Mean quote size is the mean of nonzero bid and ask sizes in parsed submitted quote commands, first within games and then across games. The two sides are separate observations. Sizes are recorded after parser clamping; subsequently rejected quotes remain submissions, while omitted sides, passes, and parsing failures contribute no quote size. This is not mean executed fill size.

Pass share in Appendix C.1 is the mean per-game fraction of turns where all parsed commands are passes; an empty command list also counts. API/parsing fallbacks can therefore contribute, so this is not exclusively deliberate inactivity.

Mean absolute inventory averages end-of-turn exposure. The quote correlations use parsed submitted two-sided quote midpoints, computing Pearson correlation within games and then averaging. These describe quoting behavior, not the accuracy of private value estimates. Final PnL decomposes into FIFO closed-position PnL and the settlement contribution of unmatched positions in the zero-fee runs (Appendix C.1).

A.5 Reproducibility

The environment rules, configurations, agent interface, prompts, and metric definitions are specified in this appendix and Appendix E. Evaluation records include per-seed final results, behavioral sum-

maries, and two 100-turn game trajectories containing observations, submitted commands, fills, and end-turn statistics.

Simulation seeds control environmental randomness but do not guarantee identical actions under stochastic model inference.

B Parameter Choices and Sensitivity

We examine six environment parameters separately: jump magnitude, persistent observation bias, fresh observation noise, ordinary diffusion, inventory capacity, and mean reversion. Table 7 gives three representative values for each. The middle value is the default setting; the lower and higher values provide contrasts that help interpret its role. These are synthetic settings for controlled comparisons, not calibrated replicas of particular real markets.

The comparisons use the six-champion roster with the same set of 20 seeds per setting. Reference values are $T = 100$, $\mu = 100$, $\kappa = 0.05$, $\sigma_\theta = 0.20$, $\sigma_w = 0.02$, $I_{\max} = 5$, $S_{\max} = 1$, limit orders only, two customers with informedness $[0.5, 0.5]$, zero initial cash and fees, and baseline inference without self-refine. Only the named environment parameter changes from its default. We reuse base and `hivol` results and evaluate eleven additional settings, each over 20 seeds. The same base results supply the middle level of all six parameter comparisons. Every setting retains all 20 games.

Table 8 gives mean PnL over 20 seeds. Table 9 reports model-level taker summaries for the low and high bias and observation-noise settings; Table 10 reports field-level behavior for the low and high diffusion, inventory-capacity, and mean-reversion settings. Their aggregation conventions differ as stated below.

Table 7: The six parameter axes. Other environment settings remain at the six-champion reference values. Each default is the middle level, not an assumption of medium difficulty.

Parameter	Lower	Default	Higher	Property varied
σ_{jump}	0	0.15	0.30	Abrupt value changes
σ_b	0	0.05	0.10	Persistent observation bias
σ_η	0.10	0.20	0.40	Fresh observation noise
σ_w	0.01	0.02	0.04	Ordinary value diffusion
$I_{\max}$	2	5	10	Inventory capacity
κ	0	0.05	0.10	Mean-reversion strength

B.1 Effect of jump magnitude

The jump scale $\sigma_{\text{jump}} \in \{0, 0.15, 0.30\}$ controls the magnitude of abrupt changes in hidden value, while their scheduled turns remain 25, 50, and 75. With $\mu = 100$, the nonzero levels correspond to jump standard deviations of 15 and 30. At zero, ordinary diffusion remains, so the value still changes between turns. The default introduces discrete repricing events without using the largest tested jump scale.

Panel (a) of Table 8 shows different responses to this change. GPT-5.5's mean PnL rises from 77.7 without jumps to 111.6 at the default and 167.7 at the higher scale; Qwen3-235B instead falls from -25.7 to -69.8 and -99.6. Gemini 3.1 Pro's means are 257.3, 212.9, and 287.2, so its response is not monotonic across these levels. Larger jumps can increase both inventory risk and the opportunities associated with a change in value; agents also receive advance information about jump direction. The results therefore describe sensitivity to abrupt repricing, not a uniform increase in task difficulty or a direct score of adaptation ability.

B.2 Effect of persistent observation bias

The bias scale $\sigma_b \in \{0, 0.05, 0.10\}$ determines the distribution of per-agent offsets drawn once and held fixed throughout each game. The corresponding standard deviations are 0, 5, and 10 price units. Setting $\sigma_b = 0$ removes persistent offsets, not the fresh observation noise. Unlike that noise, a fixed

Table 8: Mean final PnL across 20 seeds for the six parameter axes in Table 7. Panels (a)–(f) follow the same order. All default columns reuse the same base results; the highest jump scale reuses `hivol`, and the other eleven settings are additional runs. Values are rounded to one decimal; differences discussed in the text use values before display rounding. Bold marks the highest mean within each column of each panel.

(a) Jump magnitude			
Model	$\sigma_{\text{jump}} = 0$	$\sigma_{\text{jump}} = 0.15$ (default)	$\sigma_{\text{jump}} = 0.30$
GPT-5.5	77.7	111.6	167.7
Opus 4.8	115.4	114.8	99.0
Gemini 3.1 Pro	**257.3**	**212.9**	**287.2**
Sonnet 4	12.3	27.9	31.7
Qwen3-235B	−25.7	−69.8	−99.6
GLM-5.2	73.8	78.5	86.6
(b) Persistent observation bias			
Model	$\sigma_b = 0$	$\sigma_b = 0.05$ (default)	$\sigma_b = 0.10$
GPT-5.5	117.4	111.6	66.2
Opus 4.8	118.6	114.8	121.9
Gemini 3.1 Pro	**234.4**	**212.9**	**242.3**
Sonnet 4	75.4	27.9	57.0
Qwen3-235B	−5.6	−69.8	−86.1
GLM-5.2	146.0	78.5	101.6
(c) Observation noise			
Model	$\sigma_\eta = 0.10$	$\sigma_\eta = 0.20$ (default)	$\sigma_\eta = 0.40$
GPT-5.5	88.0	111.6	134.8
Opus 4.8	77.3	114.8	175.4
Gemini 3.1 Pro	**195.6**	**212.9**	**368.7**
Sonnet 4	33.3	27.9	98.8
Qwen3-235B	−13.6	−69.8	−41.0
GLM-5.2	16.2	78.5	188.4
(d) Ordinary diffusion			
Model	$\sigma_w = 0.01$	$\sigma_w = 0.02$ (default)	$\sigma_w = 0.04$
GPT-5.5	79.3	111.6	112.8
Opus 4.8	91.8	114.8	101.1
Gemini 3.1 Pro	**211.1**	**212.9**	**283.4**
Sonnet 4	36.6	27.9	54.1
Qwen3-235B	−43.1	−69.8	−25.1
GLM-5.2	103.9	78.5	144.0
(e) Inventory capacity			
Model	$I_{\max} = 2$	$I_{\max} = 5$ (default)	$I_{\max} = 10$
GPT-5.5	89.1	111.6	115.8
Opus 4.8	87.5	114.8	167.0
Gemini 3.1 Pro	**139.2**	**212.9**	**349.6**
Sonnet 4	4.3	27.9	40.8
Qwen3-235B	−44.3	−69.8	−90.6
GLM-5.2	98.5	78.5	46.9
(f) Mean reversion			
Model	$\kappa = 0$	$\kappa = 0.05$ (default)	$\kappa = 0.10$
GPT-5.5	**138.3**	111.6	126.1
Opus 4.8	80.5	114.8	127.5
Gemini 3.1 Pro	128.1	**212.9**	**204.2**
Sonnet 4	−6.5	27.9	65.0
Qwen3-235B	−88.7	−69.8	−27.2
GLM-5.2	5.2	78.5	103.2

offset cannot be removed simply by averaging more observations. This axis changes how reliable an agent's private view is over repeated interactions, rather than how quickly the market itself moves.

Panel (b) shows that this change affects models differently. GPT-5.5's mean PnL decreases from 117.4 to 111.6 to 66.2 as the bias scale increases; Qwen3-235B's decreases from -5.6 to -69.8 to -86.1. Opus 4.8's means remain much closer, at 118.6, 114.8, and 121.9. These are descriptive differences in sensitivity to persistent offsets, not proof that one model has recovered its bias more accurately.

The endpoint behavior provides additional context. GPT-5.5's taker share changes little between zero and high bias (54.7% versus 56.0%), even though its mean PnL declines. Qwen's taker share falls from 37.7% to 34.2% while its mean PnL also declines. Thus, lower performance under larger bias need not be accompanied by more liquidity taking. This comparison motivates examining how observations influence individual orders, rather than treating taker share alone as an explanation.

B.3 Effect of observation noise

The noise scale $\sigma_\eta \in \{0.10, 0.20, 0.40\}$ controls the fresh error in each private observation, with standard deviations of 10, 20, and 40 price units. The persistent bias distribution and hidden-value process remain unchanged. The default therefore provides an intermediate observation-noise scale, while the endpoints test substantially cleaner and noisier private inputs.

Panel (c) shows that noisier observations do not necessarily reduce trading PnL. GPT-5.5, Opus 4.8, Gemini 3.1 Pro, and GLM-5.2 have increasing mean PnL across the three reported levels. Sonnet 4 and Qwen3-235B do not follow this ordering. Comparing the two additional endpoint runs, five models have higher mean PnL at $\sigma_\eta = 0.40$ than at 0.10.

Trading behavior also responds differently. Between the low- and high-noise endpoints, GPT-5.5's taker share rises from 43.3% to 54.8%, whereas Gemini's falls from 15.2% to 11.8%; both have higher mean PnL in the high-noise condition. Greater noise can change other agents' quotes and the opportunities available to trade, not just the focal model's information quality. These results therefore characterize performance in different information conditions; they do not imply that noisier signals improve a model's estimates or identify a single behavioral mechanism behind the PnL changes.

Table 9: Taker share (%) at the low and high settings of persistent observation bias and observation noise. Values are mean game-level fill-record fractions over 20 eligible games, not volume-weighted shares. The PnL comparison in Table 8 additionally includes the default level of each axis.

	Persistent bias		Observation noise	
Model	$\sigma_b = 0$	$\sigma_b = 0.10$	$\sigma_\eta = 0.10$	$\sigma_\eta = 0.40$
GPT-5.5	54.7	56.0	43.3	54.8
Opus 4.8	10.9	11.4	11.2	12.0
Gemini 3.1 Pro	14.5	13.8	15.2	11.8
Sonnet 4	34.3	43.7	40.0	43.8
Qwen3-235B	37.7	34.2	43.9	38.0
GLM-5.2	11.5	10.4	12.2	10.4

B.4 Effect of ordinary diffusion

The diffusion scale $\sigma_w \in \{0.01, 0.02, 0.04\}$ controls ordinary turn-to-turn innovations in hidden value, corresponding to standard deviations of 1, 2, and 4 price units at $\mu = 100$. Scheduled jumps remain unchanged. Unlike σ_η, this parameter changes the value process itself, not the noise in the agent's private measurement.

All six models have higher mean PnL at the high endpoint than at the low endpoint (Table 8, panel d), but their responses differ in magnitude and variability. Gemini 3.1 Pro increases by 72.29, while Opus 4.8 increases by 9.26. Across the field, execution volume rises by 6.7% and pooled taker share rises by 1.34 percentage points (Table 10), while mean absolute inventory falls slightly. The

higher-PnL condition thus involves more execution, not greater average inventory exposure; these observations do not establish a universal easy-to-hard ordering.

B.5 Effect of inventory capacity

The inventory limit $I_{\max} \in \{2, 5, 10\}$ changes the capacity to carry positions, while the per-order cap remains $S_{\max} = 1$. Widening the limit from 2 to 10 raises mean PnL for Gemini 3.1 Pro, Opus 4.8, GPT-5.5, and Sonnet 4, but lowers it for GLM-5.2 and Qwen3-235B. For example, Gemini gains 210.41, whereas GLM loses 51.61. Wider capacity is therefore not uniformly beneficial.

Across the field, executed volume rises by 26.5%, cap occupancy falls from 28.32% to 1.13%, and inventory-related rejection frequency falls from 2.68% to 0.20%. These measures show that the tight cap restricts trading activity. The lower occupancy refers to each setting's own limit: absolute inventory increases from 1.029 to 2.023, so lower occupancy does not mean lower exposure. The field-level pattern alone does not explain why particular models lose PnL when capacity expands.

B.6 Effect of mean reversion

The rate $\kappa \in \{0, 0.05, 0.10\}$ controls the pull toward μ. At zero, ordinary diffusion and jumps remain but there is no restoring term. Increasing κ changes persistence and the value distribution, rather than only the quality of an agent's observations.

Five models have higher mean PnL at $\kappa = 0.10$ than at zero, while GPT-5.5 falls from 138.32 to 126.09. At zero, GPT-5.5 ranks first by mean PnL, ahead of Gemini 3.1 Pro (128.12); this mean ranking alone does not establish a statistically reliable advantage, particularly given the large variability. Across the field, stronger reversion accompanies 9.5% more execution, a 1.69-percentage-point increase in pooled taker share, and fewer inventory rejections. This is consistent with different executable trading opportunities, but does not isolate the mechanism behind each model's performance change.

Field-level behavioral aggregation. Table 10 pools the six models in each setting. PnL and volume average 120 model-games; volume counts executed model-side contract units. A model-to-model fill contributes units to both participating models. Taker share divides taker units by all executed model-side units; unlike Table 9, it is not an equal average of game-level percentages. Mean absolute inventory and cap occupancy use 12,000 post-turn model observations per setting. Rejection rates divide next-turn-visible inventory-limit rejects by attempted bid/ask sides on turns 1–99; final-turn submissions are excluded because their feedback is not observed. Behavioral summaries are available only for the low and high settings, so this comparison excludes the default level.

Table 10: Field-level behavior under varying diffusion, inventory capacity, and mean reversion. Each setting has 20 games, six models, and 100 turns. Rejection cells give percentage and numerator/denominator. All other percentages follow the aggregation definitions in the text. These field-level summaries do not identify model-specific mechanisms.

Setting	PnL	Taker (%)	Volume	$\lvert I \rvert$	At limit (%)	Rejects % (count/attempts)
$\sigma_w = 0.01$	80.0	26.69	69.15	1.805	11.15	1.07 (153/14305)
$\sigma_w = 0.04$	111.7	28.03	73.79	1.777	9.15	1.08 (165/15329)
$I_{\max} = 2$	62.4	26.37	65.67	1.029	28.32	2.68 (378/14095)
$I_{\max} = 10$	104.9	30.27	83.04	2.023	1.13	0.20 (33/16362)
$\kappa = 0$	42.8	26.50	68.05	1.640	7.91	1.27 (185/14593)
$\kappa = 0.10$	99.8	28.19	74.53	1.787	8.95	0.98 (150/15357)

C Additional Results and Game Examples

Aggregate outcomes identify differences between models but do not show the individual decisions behind them. We first compare participation, liquidity taking, and PnL accounting, then examine

two games through their observations, orders, and fills. The games are illustrative trajectories, not randomly sampled evidence of a universal strategy or additional intervention trials.

C.1 Behavioral differences across models

We examine the 14-model base field (20 seeds) to connect model rankings with trading behavior. Its mean PnL values match the base column of Table 3. The field-specific configuration, including $\kappa = 0$, is documented in Appendix A. The two games discussed below, Case 1 and Case 2, come from this same field.

Participation is different from taking liquidity. Table 11 separates how often a model passes from how often its fills take liquidity. Gemini combines a low taker share (9.3%) with few recorded pass turns (6.65%). It therefore participates frequently without usually accepting another trader's quote for immediate execution. GLM-5.2 has a similar profile (9.7% taker, 6.25% pass); this combination is not unique to Gemini. Opus 4.8 also has a low taker share (6.4%) but passes more often (19.70%). A pass does not cancel resting orders, and a non-pass action need not submit both a bid and an ask, so these measures describe different aspects of activity. Neither low taker share nor frequent participation alone establishes a better strategy: a resting quote can also trade at an unfavorable price.

Where the accounting difference appears. The accounting separates profit on closed positions from the settlement contribution of positions still open at the end. Closed-position PnL uses first-in, first-out matching of buys and sells. Closed-position PnL is not a pure measure of bid–ask spread capture: it includes closed trades made as either maker or taker and can reflect price changes between entry and exit. Table 12 compares Gemini with Opus 4.8 in the same 14-model field. Their mean PnL difference is 101.6, of which 94.1 (about 93%) lies in the closed-position component. Thus, most of this accounting difference is not a terminal gain on remaining inventory. It does not, however, identify which trading decisions caused the gap. Opus 4.8 is the runner-up in the champion ranking, not in this 14-model base field.

How closely do quotes follow the private observation? Gemini's quote midpoint correlates more strongly with hidden value (0.5) than with its current private observation (0.3). GPT-5.5 shows the reverse ordering (0.3 versus 0.4), as do Qwen3-235B and Gemini 2.5 Flash. These are descriptive associations of submitted quotes, not measures of the accuracy of models' beliefs: quote placement also serves inventory and execution objectives. In particular, GPT-5.5 remains profitable in this base field ($+67$), despite a much higher taker share (60.8%). The following two case studies examine actual orders and fills to make these behavioral comparisons concrete.

C.2 Case 1: from observations to executed trades

Case 1 follows Gemini 3.1 Pro, Opus 4.8, and GPT-5.5 in the same 14-model game, with the configuration described in Appendix C.1. All three face the same hidden-value path but receive different private observations and have different trading histories and inventories.

The same jump, different observations and orders. At turn 25, the hidden value rises from 80.69 to 99.45. At turn 26 it remains nearby, at 98.87, but the private observations range from 69.10 to 115.27 (Table 13). Gemini buys at 87 in both turns, increasing inventory from one to three contracts. Opus submits an 84/87 quote at turn 25; a customer buys its ask at 87, leaving it short one contract. At turn 26 it submits a buy-only quote at 85 but receives no fill. GPT-5.5 buys at 86 at turn 25, then sells at 86 at turn 26 after its private observation falls from 97.42 to 69.10. Its recorded explanation explicitly mentions the lower observation and its desire to reduce inventory risk. The immediate buy–sell pair earns zero before fees, which are zero here; it does not demonstrate a realized loss. Instead, the example shows how a sharp change in a noisy observation can accompany a reversal in trading, even when the hidden value barely changes. It also shows why intentions must be checked against fills: a submitted buy order need not execute, and a two-sided quote can fill only on the sell side.

Table 11: Trajectory-level behavior, base config (14-model all-in-one field, 20 seeds), sorted by tot. tot=PnL=mm+pos; mm=closed-position PnL, pos=remaining-position settlement PnL; taker%=taker fill share; pass%=recorded pass turns divided by total turns, averaged across seeds; |inv| =mean absolute inventory; $\rho(\text{mid}, \theta)/\rho(\text{mid}, \text{raw})$ = correlation of the agent's quote mid with the hidden value vs. its own noisy read. Rounded PnL components may not sum exactly.

model	tot	mm	pos	taker%	pass%	|inv|	$\rho(\cdot, \theta)$	$\rho(\cdot, \text{raw})$
Gemini 3.1 Pro	**244**	**235**	+8	9.3	6.65	1.5	0.5	0.3
GLM-5.2	199	193	+6	9.7	6.25	1.4	0.3	0.2
Opus 4.7	179	179	+0	7.9	18.20	1.1	0.3	0.2
Opus 4.6	171	170	+0	5.2	28.85	0.9	0.3	0.2
Opus 4.8	142	141	+1	6.4	19.70	0.6	0.3	0.2
GPT-5.5	67	60	+6	60.8	1.35	1.3	0.3	0.4
Gemini 2.5 Pro	59	59	−0	36.7	11.00	1.0	0.3	0.4
GPT-4.1	29	16	+13	27.6	33.55	1.0	0.2	0.3
GPT-5.2	24	23	+0	44.0	6.95	0.8	0.2	0.2
Sonnet 4	23	16	+7	36.5	40.65	1.5	0.3	0.4
GPT-5.4	21	21	−0	36.5	15.65	0.8	0.3	0.3
Gemini 3.5 Flash	14	12	+2	55.4	7.65	1.5	0.3	0.3
Qwen3-235B	−42	−40	−2	43.5	15.20	2.4	0.4	0.5
Gemini 2.5 Flash	−76	−76	−1	71.8	3.50	1.1	0.3	0.5

Table 12: PnL accounting comparison between Gemini 3.1 Pro and Opus 4.8 in the 14-model base field (20 seeds). Differences are computed from unrounded means; displayed values are rounded independently. The closed-position component includes both maker and taker trades.

	total	mm (closed positions)	pos (remaining positions)
Gemini 3.1 Pro	+243.7	+235.3	+8.4
Opus 4.8	+142.0	+141.3	+0.8
difference	+101.6	+94.1	+7.6

From a local decision to a whole-game outcome. Figure 2 places this episode in the full trajectory. The logged market price broadly follows the hidden value but can lag or overshoot it; their correlation in this game is 0.88 and their mean absolute difference is 4.93. Gemini's inventory ranges from -5 to $+4$, Opus's from -1 to $+2$, and GPT-5.5's from -2 to $+4$. Their final PnLs are $+180.2$, $+146.0$, and -70.2, respectively. These final outcomes reflect the full trading sequence.

C.3 Case 2: adjusting quotes under a biased observation

Case 2 uses the same configuration as Case 1 (Appendix C.2) and illustrates how a model revises its quotes when its private observation conflicts with trading information. Gemini 3.1 Pro's actual persistent bias is $b_j = +13.13$, or about 2.6 bias standard deviations. Its private observations average 12.41 above the hidden value across the 100 turns. Neither the bias nor the hidden value is directly

Table 13: Decisions around the first jump in Case 1. s is the agent's private observation; quotes show submitted bid/ask, with — for an omitted side. Fills are the actual executions in that turn (M = maker, T = taker); all sizes are one. Inventory is measured after the turn, including fills of any older orders. Hidden values are 99.45 at turn 25 and 98.87 at turn 26.

Turn	Model	s	Bid/ask	Executed fills	End inventory
25	Gemini 3.1 Pro	116.07	105/115	Buy 87 (T)	+2
25	Opus 4.8	102.19	84/87	Sell 87 (M)	−1
25	GPT-5.5	97.42	88/100	Buy 86 (T)	+1
26	Gemini 3.1 Pro	115.27	100/115	Buy 87 (T)	+3
26	Opus 4.8	86.53	85/—	None	−1
26	GPT-5.5	69.10	—/85	Sell 86 (T)	0

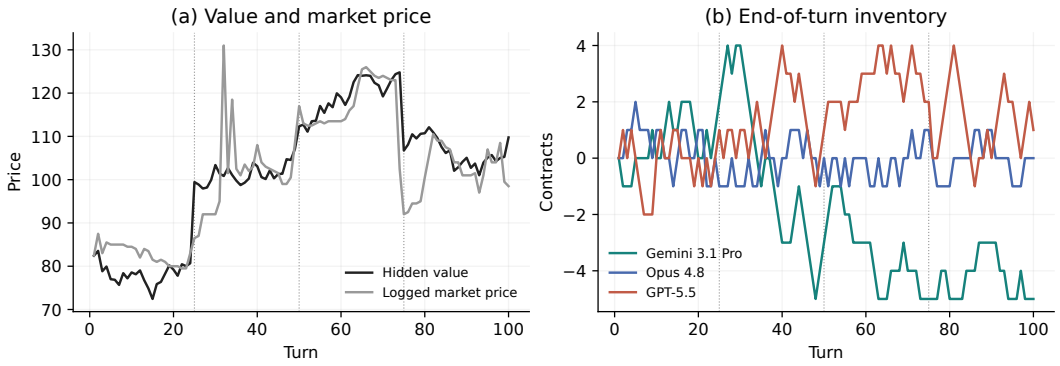

Figure 2: Case 1: a 14-model game with three focal models. **(a)** Hidden value and logged market price. **(b)** End-of-turn inventory. Dotted lines mark jump turns 25, 50, and 75.

available to the model. We compare its recorded explanations with submitted orders and executed fills.

A bias hypothesis followed by lower quotes. At turn 1, Gemini notes that it needs to observe the flow to assess bias. It submits 99/101 and completes a buy at 99 and a sell at 101. At turn 2, its observation rises to 110.34. Its explanation proposes buying an ask at 106, and it submits 106/114. The fill record, however, identifies Gemini as the *maker* when Sonnet 4 sells to its resting bid at 106. The purchase is 25.22 above the contemporaneous hidden value of 80.78, illustrating that an unfavorable execution need not be a taker trade.

At turn 3, Gemini writes: "The model seems to have a persistent high bias (perhaps +10 to +20)." It submits a cancellation of its old 114 ask and a lower 78/87 quote, both sides of which trade. At turn 5, despite a higher observation of 114.48, it submits 75/85 and sells one contract at 85. These commands show an adjustment away from simply centering quotes on the latest private observation (Table 14).

Table 14: Early decisions by Gemini 3.1 Pro in Case 2. Hidden value θ_t is included for offline comparison only. Quotes are submitted bid/ask; all orders and fills have size one. M/T denotes the focal model's executed role. Selected turns are not consecutive, so inventory changes also reflect omitted turns.

Turn	θ_t	$s_{j,t}$	Bid/ask	Executed fills	End inventory
1	82.08	100.39	99/101	Buy 99, sell 101 (both M)	0
2	80.78	110.34	106/114	Buy 106 (M)	+1
3	82.03	95.78	78/87	Buy 78, sell 87 (both M)	+1
5	81.72	114.48	75/85	Sell 85 (M)	+1
6	82.26	122.84	85/95	Buy 85 (T)	+2

Adjustment is not exact recovery of the bias. By turn 5, Gemini describes the bias as "around 25-30 points," substantially above the true $+13.13$. At turn 6 it uses that estimate to justify a higher 85/95 quote after another high observation. The trajectory therefore shows a model questioning and revising its use of private information, not accurately recovering a fixed bias once and for all. Moreover, its own orders help shape the market information it subsequently observes.

Subsequent performance. Gemini finishes with a final PnL of $+470$ and zero inventory, compared with a 14-model mean of $+107.7$. Later fills also matter: at turn 49, a customer sells to its resting bid at 10, far below that turn's hidden value of 74.34. Thus, the final profit cannot be attributed solely to the early bias hypothesis. This example illustrates information use and quote adjustment.

Table 15: External abstention rates from the Opus 4.8 LLM judge for nine models. The Spearman rank correlation between arena taker share and abstention rate is $\rho = -0.70$. Evaluation details are given in the text.

Model	Arena taker (%)	Abstention (%): LLM judge
GPT-5.5	72.6	42.3
GPT-5.4	48.9	44.3
Gemini 2.5 Pro	46.5	45.5
Gemini 3.5 Flash	28.1	42.7
Opus 4.7	19.3	47.8
Qwen3-235B	17.0	47.7
GLM-5.2	16.4	46.1
Gemini 3.1 Pro	15.6	44.5
Opus 4.8	15.2	52.4

D Additional Instrument Experiments

This appendix documents the over-confidence analysis and the state-tracking and planning comparisons. The analyses combine intervention outcomes and observational trajectory evidence to examine model behavior, rather than assign a single score to each capability.

D.1 Over-confidence: intervention and external evaluation

Intervention protocol. The reference run is the 14-model `rich` field. For each of the nine focal models in Table 4, a separate run enables self-refine only for that model. All use the same 10-seed subset specified in Appendix A.4. The configurations differ only in which model receives self-refine; the scenario and roster match. The implementation and added instruction are documented in Appendix A.2. The task-generic instruction explicitly asks the model to check recency over-reaction, while its surrounding context retains the trading task. Thus, it is a specified intervention on the decision process, not an instruction-free test of spontaneous self-correction.

For GPT-5.5, mean PnL changes from -164.93 to $+54.58$, a gain of 219.52 (rounded to $+220$ in the main table). Its rank by mean PnL changes from 14th to 2nd within the corresponding fields.

External evaluation and detection. The external analysis uses abstention on unanswerable questions as a behavioral comparison outside the arena. The answering prompt is the question text alone as a single user message. We evaluate all models on the same 912 question records selected from `should_abstain` items. Abstention rates are calculated over non-empty responses with valid LLM judgments.

The LLM detector uses Opus 4.8 as the requested judge model.

Table 15 reports all nine models. The Spearman rank correlation between arena taker share and abstention rate is $\rho = -0.70$: models that take liquidity more often tend to abstain less on unanswerable questions. This is a cross-task behavioral comparison; self-refine is not applied in the external evaluation.

D.2 State tracking

We study state tracking by examining how access to past observations and events affects trading decisions under different observation interfaces. This comparison studies memory use in trading decisions, not the accuracy of a model's unobserved value estimate. Access to history need not imply that a model uses it effectively.

The experiment places two seats of the same model in one game. Seat A removes retained dialogue and scratchpad updates and receives an explicit no-memory instruction; seat B retains the baseline procedure. Both still receive the current private signal, cash, inventory, own resting orders, and previous-turn events. The configured output/reasoning budget is unchanged. The outcome is $\Delta = \mathrm{PnL}_A - \mathrm{PnL}_B$, so a negative mean favors the baseline seat.

In book-shown mode, inputs include the anonymized book snapshot, public trades, and the viewer's own order-status events. Book-hidden mode replaces the snapshot with public accepted orders, trades, successful cancels, and expirations carrying stable order IDs. Own rejections and current resting orders remain visible. A baseline agent can accumulate these events to reconstruct the book, whereas the last turn's events alone do not reveal every older resting order.

The nine book-hidden memoryless-versus-baseline comparisons use $T = 100$, $I_{\max} = 5$, $S_{\max} = 10$, all order types, $\kappa = 0.05$, one customer with informedness 0.6, and 10 seeds. Table 16 reports PnL differences for both observation interfaces.

Table 16: Within-model duel PnL differences (memoryless minus baseline), averaged over 10 seeds. Negative means favor baseline; values are rounded to integers.

Model	ΔPnL (book shown)	ΔPnL (book hidden)
Gemini 3.5 Flash	−1205	−760
Gemini 3.1 Pro	−598	−385
GLM-5.2	+1	−210
Opus 4.8	+10	−37
GPT-5.5	+180	−17
GPT-5.4	+43	−4
Gemini 2.5 Pro	+179	+7
Qwen3-235B	+128	+16
Opus 4.7	+21	+35

With the book hidden, baseline has substantially higher mean PnL for Gemini 3.5 Flash, Gemini 3.1 Pro, and GLM-5.2. GPT-5.5's difference is much smaller (-17), while Opus 4.7, Qwen3-235B, and Gemini 2.5 Pro have positive means. These outcomes identify different performance responses to removing memory, not a uniform benefit from retaining it. Small signed means should not be interpreted as reliable advantages without considering variation across games.

Across the two observation interfaces, Gemini 3.5 Flash favors baseline in both conditions, whereas GLM-5.2 changes from a near-zero difference with the book shown ($+1$) to -210 with it hidden. These PnL comparisons alone do not distinguish calibration, inventory-management, or order-reconstruction mechanisms.

D.3 Planning

Planning requires present actions to account for future consequences. Inventory and resting orders provide such links, but their presence alone does not establish that looking further ahead improves performance. Larger total PnL in a longer game can also result from more trading opportunities.

The study uses 200-turn, six-seat same-model games with one myopic seat A and five baseline seats B–F, 20 seeds, limit orders, $S_{\max} = 1$, $I_{\max} = 5$, two customers with informedness $[0.5, 0.5]$, $\kappa = 0.05$, and jumps every 25 turns through turn 175. Myopic removes explicit total-horizon cues and requests greedy current-turn optimization (Appendix A.2), while retaining turn index, history, scratchpad, observations, events, and jump warnings. It changes both horizon information and the instructed objective, rather than mechanically eliminating planning.

Let $\mathcal{S}$ denote the set of 20 seeds. For each seed $s \in \mathcal{S}$, the final-PnL comparison is

$$d_s = \frac{1}{5} \sum_{j \in \{B,C,D,E,F\}} \mathrm{PnL}_{j,s} - \mathrm{PnL}_{A,s}, \qquad \Delta_{\mathrm{plan}} = \frac{1}{20} \sum_{s \in \mathcal{S}} d_s.$$

Positive values favor baseline. The unit of replication is a game, not each of the five interacting baseline seats. For each model, the normal-approximation 95% confidence interval is $\Delta_{\mathrm{plan}} \pm 1.96\, s_d/\sqrt{20}$, where s_d is the sample standard deviation of the 20 game-level differences. These pointwise intervals concern final PnL, not its closed-position component, and are not simultaneous guarantees across all models.

Table 17 reports the planning comparison for 11 models from the main evaluation roster. GPT-5.4, GPT-5.2, and Gemini 3.1 Pro favor baseline, with intervals excluding zero; GPT-5.5 favors

Table 17: Planning comparison: mean baseline-minus-myopic final PnL and normal-approximation 95% confidence intervals over 20 games per model. Positive differences favor baseline.

Model	Δ_{plan}	95% interval
GPT-5.5	-767.4	$[-1051.1, -483.7]$
GPT-5.4	$+335.0$	$[+228.3, +441.8]$
GPT-5.2	$+347.2$	$[+194.4, +500.0]$
GPT-4.1	$+8.0$	$[-173.6, +189.6]$
Sonnet 4	$+100.8$	$[-95.7, +297.3]$
Opus 4.8	$+15.1$	$[-4.3, +34.5]$
Opus 4.7	-9.9	$[-45.3, +25.5]$
Opus 4.6	-11.0	$[-36.7, +14.6]$
Gemini 3.1 Pro	$+128.6$	$[+38.3, +218.8]$
Gemini 2.5 Pro	-52.6	$[-204.9, +99.7]$
Gemini 2.5 Flash	-145.9	$[-292.5, +0.8]$

myopic. The remaining intervals include zero. Thus, the same myopic instruction has different performance consequences across models, rather than a uniform effect that can be interpreted as a planning-capability score.

An illustrative GPT-5.5 episode at turn 7 contrasts a baseline seat describing a jump-related trading plan and crossing quotes with a myopic seat choosing a tight passive two-sided quote. This is consistent with different trading responses to the instructions, but does not show that the same mechanism explains all seeds or other models' results.

E Prompt Text

E.1 Baseline system prompt

The following baseline prompt is instantiated for the planning configuration: 200 turns, six makers, two uninformed customers, limit orders, $S_{\max} = 1$, $I_{\max} = 5$, zero fees, and book observation. Generic command examples can exceed the configured size cap; the stated parser clamping applies.

```
You are a market maker on a simulated exchange.

## The Contract

You are a market maker quoting a cash-settled futures contract. At
expiry (the final turn) it settles to the underlying's fair value,
which you never see directly.

Each turn your desk's pricing model prints one number:

- `model_estimate`: the model's estimate of the underlying's fair
value this turn. It is noisy, and beneath the noise it carries a
persistent systematic bias (a fixed high or low lean).

## Price Dynamics

The underlying value follows a jump-diffusion: most turns it only
drifts a little, but once in a while it jumps. The turn before a jump,
your state carries a `jump_warning` with the jump's direction (up or
down) -- never its size.

## How Trading Works

When your quote crosses a resting order (your bid >= a resting ask, or
your ask <= a resting bid), a trade executes at the resting order's
price. Orders at the same price level are matched FIFO.

The game lasts 200 turns with 6 market makers (including you). Each
```

```
turn, after the market makers act, 2 other counterparties each send a
1-contract market order that hits your best bid or lifts your best ask
-- uninformed customers trading at random.

## Your State Each Turn

Each turn you get a JSON state: a snapshot of where you stand now,
plus an `events` list of what happened since your last turn, in time
order.

Snapshot:
- `cash`, `inventory` -- your current position.
- `model_estimate` -- this turn's pricing-model print (see The
Contract).
- `your_open_orders` -- your resting orders: `order_id`, `side`,
`price`, `size`.
- `order_book` -- every resting order on each side: `order_id`,
`price`, `size`, `mine` (true if yours). The `order_id` is stable, so
you can track an order across turns.

`events` -- time-ordered since your last turn; each has a `type`:
- `trade`: an execution, with these fields:
  - `price`, `size` -- execution price and quantity.
  - `aggressor` -- the taker's side, `"buy"` or `"sell"`.
  - `order_id` -- the resting (maker) order that was hit (public). If
  you are the taker, this is the counterparty order you hit (the one
  you saw in `order_book`); if you are the maker, it is your own
  order. The aggressor has no public id.
  - `you` -- `null` if you were not in this trade, else `{"role":
  "maker"|"taker", "side": "buy"|"sell", "your_order_id": <id>}`,
  where `your_order_id` is YOUR order in this fill (same id whether it
  took or rested).
- `accepted`: your order was accepted with an `order_id` (`side`,
`price`, `size`, `order_type`). It then rests on the book; a
marketable order is accepted then trades, and any resting remainder
keeps this same `order_id`. How much rested = accepted size minus your
fills this turn (also shown in `your_open_orders`).
- `expired`: an unfilled take-order remainder was dropped (not
rested). `side`, `size`, `reason`.
- `reject`: a side/order was rejected. `side` (`"bid"`/`"ask"`/`null`)
+ `reason`.
- `cancel`: your cancel result. `order_id`, `ok`, and `reason` if it
failed.

## Available Commands

- `quote <bid>x<bid_size> <ask>x<ask_size>` -- post a passive
two-sided quote. Each side is `price`x`size`, e.g. `quote 99x3 102x1`.
  - bid = highest price you will BUY at; ask = lowest price you will
  SELL at
  - use `-1` for a side to skip it (e.g. `quote 99x3 -1`; `quote -1
  -1` posts nothing)
- `cancel <order_id> [<order_id> ...]` -- cancel your resting orders
by ID
- `pass` -- do nothing this turn

These apply to every order (quotes and take orders):
- size is 1..1 contracts (clamped into range)
- prices snap to the nearest tick of 1.0
- a side that would breach the position limit is rejected (that side
only)

Orders rest on the book across turns until filled or cancelled.
Position limit: inventory plus pending orders stays within +/-5. Cash
may go negative; only the position limit constrains you. You may
```

```
submit at most ONE order (a quote or a take order) plus cancels per
turn.

## Settlement

    Final PnL = (Cash - Starting Cash) + Inventory * SettlementValue

You start with 0 cash (so Cash in your state is already your cash
change). Your objective is to maximize your total PnL over the game;
the trading strategy is entirely up to you.

## Output Format

Respond with ONLY valid JSON, no other text:

{
  "thoughts": "your reasoning about value, market state, and
  strategy",
  "scratchpad": "optional -- overwrite scratchpad with this text",
  "scratchpad_append": "optional -- append to the existing
  scratchpad",
  "commands": ["cancel 3 5", "quote 19x2 23x1"]
}

All agents act simultaneously on the same end-of-previous-turn book;
actions then execute in a random order.

## Memory

Your conversation history is limited to the last 20 turns. Use the
scratchpad fields in your JSON output to persist anything you want to
carry across turns:
- "scratchpad": "..." -- overwrite the entire scratchpad
- "scratchpad_append": "..." -- append to the existing scratchpad Both
are optional. The scratchpad is injected into your system prompt every
turn.

### CURRENT SCRATCHPAD
(empty)
```

Numeric fields are rendered from the configuration. Flow mode replaces the book snapshot with public order-flow events. Rich-action configurations add post-only quote suffixes and market, IOC, and FOK commands; nonzero fees add a fee clause. The baseline explains noise and persistent bias without prescribing a strategy. Memoryless, myopic, and self-refine modifications are documented separately in Appendix A.2.