CATTLE TRADE: A MULTI-AGENT BENCHMARK FOR LLM BLUFFING, BIDDING, AND NEGOTIATION

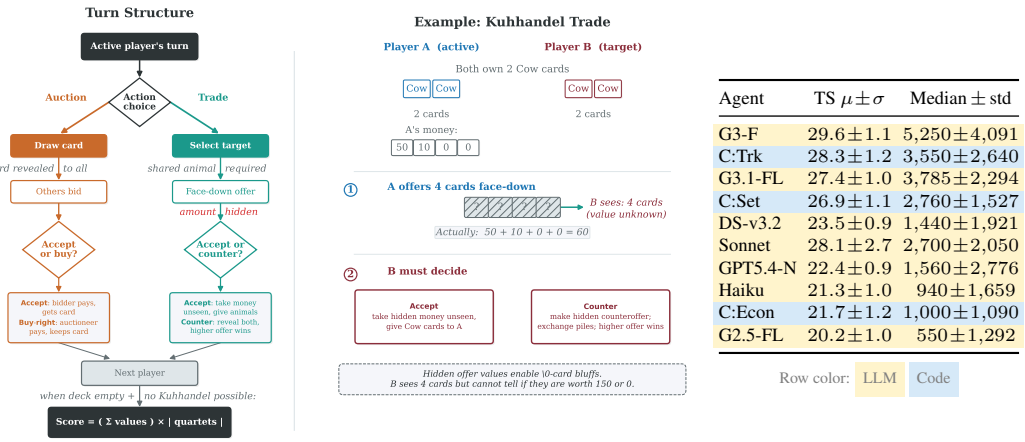
Robert Müller* **Clemens Müller***

ABSTRACT

Cattle Trade[1] is a multi-agent benchmark based on the card game Kuhhandel combining auctions, face-down bargaining, and discrete money into a single decision space across 50+ turn games. The benchmark logs every bid, offer, and card selection, enabling behavioural analysis beyond win rates. Evaluating seven cost-efficient LLMs and three deterministic code agents across 232 games surfaces systematic failures in resource-constraint adherence, self-bidding avoidance, and opponent modelling. Two heuristic code agents each outperform four of seven LLMs, and spending efficiency predicts final ranking better than spending volume.

1 INTRODUCTION

LLM-based agents are increasingly deployed in multi-agent settings, from negotiation (Lewis et al., 2017) to cooperative tool use (Hong et al., 2024). Standard benchmarks such as MMLU (Hendrycks et al., 2021) and HumanEval (Chen et al., 2021) measure knowledge and code generation but tell us little about strategic reasoning under uncertainty and adversarial incentives. Game-based benchmarks for social deduction (Light et al., 2023; Xu et al., 2023), game-theoretic scenarios (Duan et al., 2024), and social reasoning (Pan et al., 2024) each test individual competencies in isolation rather than requiring their combination within a single environment.



Agent	TS $\mu \pm \sigma$	Median $\pm$ std
G3-F	29.6 ± 1.1	$5{,}250 \pm 4{,}091$
C:Trk	28.3 ± 1.2	$3{,}550 \pm 2{,}640$
G3.1-FL	27.4 ± 1.0	$3{,}785 \pm 2{,}294$
C:Set	26.9 ± 1.1	$2{,}760 \pm 1{,}527$
DS-v3.2	23.5 ± 0.9	$1{,}440 \pm 1{,}921$
Sonnet	28.1 ± 2.7	$2{,}700 \pm 2{,}050$
GPT5.4-N	22.4 ± 0.9	$1{,}560 \pm 2{,}776$
Haiku	21.3 ± 1.0	$940 \pm 1{,}659$
C:Econ	21.7 ± 1.2	$1{,}000 \pm 1{,}090$
G2.5-FL	20.2 ± 1.0	$550 \pm 1{,}292$

Row color: LLM Code

Figure 1: **(a) Turn structure and Kuhhandel trade example.** Each turn the active player auctions a drawn card or initiates a face-down trade with an opponent sharing the same animal type; card count is visible but offer values are hidden, enabling $0-card bluffs. **(b) Tournament ranking** over 88 canonical games (7 LLMs + 3 code agents), sorted by conservative TrueSkill $\mu_c = \mu - 3\sigma$ (descending). Row tint distinguishes LLMs from deterministic code agents. Two code-agent heuristics outperform four of seven LLMs.

*Equal contribution. Correspondence to: robert@aganthos.com

[1] *Kuhhandel* (English: *You're Bluffing!*) is a trademark of Ravensburger. This work is not affiliated with Ravensburger.

We introduce CATTLE TRADE, a benchmark based on the card game Kuhhandel that integrates (i) competitive auctions with a buy-right mechanism coupling bidding strategy to future liquidity, (ii) hidden-offer bilateral trades that create bluffing incentives, (iii) discrete money with no-change payments that impose resource constraints, and (iv) a multiplicative scoring rule that makes portfolio composition consequential. Games last 50–60 turns with four players. We evaluate seven cost-efficient LLMs and three deterministic code agents across 232 games in two formats: pure-LLM tournaments and mixed games where each LLM faces three code agents.

Two code-agent heuristics (TrackerAgent and SetRaceAgent) each outperform four of seven LLMs. Only Gemini 3 Flash clearly outperforms all code-agent baselines (72.7% wins, μ_c=26.2); Gemini 3.1 Flash Lite (43.2% wins, μ_c=24.3) beats SetRace and EconomyAgent but ranks below TrackerAgent. Behavioural analysis surfaces failures absent from the code agents: overbidding, self-bidding spirals, and bankrupt trade initiation.

Our contributions:

- A multi-agent benchmark combining auctions, bargaining, deception, and resource constraints with structured decision logging and configurable game parameters.
- A behavioural analysis suite profiling strategic play via spending efficiency, bluff rates, phase-dependent bid adaptation, self-bidding rates, and buy-right patterns.
- Empirical results across 232 games showing that spending efficiency and resource discipline predict success, while overbidding, self-bidding spirals, and undisciplined bluffing characterise failure.

2 RELATED WORK

Standard benchmarks such as MMLU (Hendrycks et al., 2021), BIG-bench (Srivastava et al., 2023), and HumanEval (Chen et al., 2021) measure knowledge and reasoning but do not test multi-agent incentives, hidden information, or long-horizon resource allocation. AgentBench (Liu et al., 2024) extends evaluation to interactive tool-use settings but lacks adversarial strategic reasoning.

Game-based LLM benchmarks each emphasize a subset of strategic skills. GTBench (Duan et al., 2024) covers game-theoretic scenarios but uses short-horizon interactions. AvalonBench (Light et al., 2023) and Werewolf (Xu et al., 2023) probe deception and cooperation but lack economic constraints. MACHIAVELLI (Pan et al., 2024) provides long-horizon continuous scoring but pits a single agent against scripted dynamics. Suspicion-Agent (Guo et al., 2024) studies deception in an imperfect-information board game; GameBench (Costarelli et al., 2024) broadens game coverage but simplifies economic structure; lmgame-Bench (Hu et al., 2026) proposes a Gym-style interface with scaffolds for perception and memory. To our knowledge, no existing benchmark requires agents to combine auctions, hidden-offer bargaining, bluffing, and discrete resource management within a single game.

Jia et al. (2025) analyze strategic reasoning through behavioral game theory, GAMEBoT (Lin et al., 2025) decomposes game reasoning into interpretable sub-problems, and ALYMPICS (Mao et al., 2025) evaluates agents in auction-like settings. Pluribus (Brown & Sandholm, 2019) and CICERO (Bakhtin et al., 2022) achieve strong play in poker and Diplomacy, respectively, but rely on domain-specific search and self-play rather than general-purpose language modeling. Work on neural negotiation (Lewis et al., 2017; Gandhi et al., 2023) and deception (Hagendorff, 2024) addresses dialogue-level strategy but not the combination of hidden offers, discrete money, and multiplicative scoring that Kuhhandel requires.

CATTLE TRADE concentrates these pressures into a single environment: a multi-agent imperfect-information game with repeated auctions and bargaining, bluffing incentives, discrete liquidity constraints, and continuous scoring over 50–60 turns. This enables both head-to-head ranking (via TrueSkill) and mechanism-level behavioral diagnosis (e.g., overbid frequency as a resource-discipline metric, self-bidding rate as a situational-awareness metric).

3 CATTLE TRADE

3.1 THE KUHHANDEL GAME

Kuhhandel (English: *You're Bluffing!*) is a 3–5 player card game designed by Rüdiger Koltze (Ravensburger, 1985). Players collect quartets (complete sets of 4 cards) of animal types, scored by a multiplicative formula that rewards both the value and number of completed sets. The game uses 40 animal cards (10 types $\times$ 4 each) and 55 money cards of fixed denominations. Figure 1 illustrates the turn structure and trade mechanic; Table 1 compares CATTLE TRADE to existing game-based LLM benchmarks. Kuhhandel combines auctions, bargaining, bluffing, and resource management within a single game over 50–60 turns.

Benchmark	Players	Info	Bargain	Bluff	Econ.	Scoring	Turns
GTBench	2	Mixed	✓	✗	✗	Binary	1–10
AvalonBench	5–10	Imperfect	✗	✓	✗	Binary	5–10
Werewolf	5–18	Imperfect	✗	✓	✗	Binary	5–15
MACHIAVELLI	1	Partial	✗	✗	✗	Continuous	50+
Suspicion-Agent	3–6	Imperfect	✗	✓	✗	Binary	10–20
CATTLE TRADE	3–5	Imperfect	✓	✓	✓	Continuous	50–60

Table 1: Comparison of game-based LLM benchmarks. Bargain: explicit bilateral offer/counter mechanic. Bluff: deception as core mechanic. Econ.: resource management required. CATTLE TRADE combines these properties among the benchmarks we surveyed.

Each animal type has a quartet value: Chicken (10), Goose (40), Cat (90), Dog (160), Sheep (250), Goat (350), Donkey (500), Pig (650), Cow (800), and Horse (1000). The final score is:

$$\text{Score} = \left(\sum_{q \in \mathcal{Q}} v_q \right) \times |\mathcal{Q}| \tag{1}$$

where $\mathcal{Q}$ is the set of completed quartets and v_q is the value of quartet q. This multiplicative structure means three mid-value quartets (e.g., 1,250 $\times$ 3 = 3,750) outscore two top quartets (1,800 $\times$ 2 = 3,600), rewarding breadth over concentration and making opponent modeling critical.

The game uses discrete, non-fungible money cards (denominations 0, 10, 50, 100, 200, 500) with no change given: paying 60 with a 100 card costs 100. Each player starts with 90 coins (four 10s, one 50, two 0s). The 0 cards exist specifically for bluffing in Kuhhandel trades: an opponent who receives a face-down offer of three cards cannot distinguish 150 coins from zero. This information asymmetry is the foundation of the game's deception mechanics.

On each turn, the active player draws an animal card and auctions it. The auctioneer either accepts the highest bid or exercises a *buy-right*, paying the highest bid amount to keep the card. If no one bids, the auctioneer receives the card for free. The buy-right creates a strategic dilemma: bidding high may win the card but risks enriching the auctioneer if they exercise buy-right. If a bidder overbids their holdings, the engine reveals their wealth to all players and restarts the auction.

Instead of auctioning, the active player may initiate a Kuhhandel trade with any opponent who shares an animal type. The initiator makes a face-down offer using any money cards, including 0 bluffs. The opponent may accept (taking the money sight-unseen and surrendering their animals) or counter with their own face-down offer; the higher offer then wins all contested animals. Bluffing succeeds only if the opponent accepts without seeing the offer; any positive counter defeats a 0 bluff. This mechanism is the strategic core of the game, requiring calibrated risk assessment: overpaying wastes resources, but underbidding loses cards.

Donkey cards trigger money distribution to all players when drawn (50, 100, 200, and 500 coins for the first through fourth donkey), injecting liquidity into the game economy and creating phase transitions: early games are cash-constrained, while late games after multiple donkey payouts allow larger bids and trades. The donkey payout schedule is public knowledge, so agents can anticipate liquidity changes.

3.2 BENCHMARK DESIGN

The benchmark consists of a game engine, an LLM agent framework, and an evaluation system. We implement the complete Kuhhandel rules in Python with three auction modes: canonical (simultaneous-intent bidding faithful to tabletop play), fast (single sealed bid), and legacy (sequential). The engine validates all actions, handles exact money card accounting, and logs complete state transitions.

Agents receive natural language observations (hand, visible cards, money, legal actions) and respond with a structured JSON action specifying exact bid amounts for auctions and exact card compositions for Kuhhandel offers (e.g., `offer_cards: [50, 10, 0]`), giving models direct control over bluff construction rather than delegating card selection to a heuristic. The framework supports three memory modes: none (stateless), events (recent $\sim$10 events), and full (events plus an agent-maintained scratchpad updated each turn via an additional model call). Each agent uses the `standard` character prompt (minimal rule description) for fair comparison. We use TrueSkill (Herbrich et al., 2006) for competitive rating, which provides Bayesian skill estimates (μ, σ).

The framework handles malformed actions via multi-stage JSON fallback and retries truncated responses. In canonical mode, the engine permits overbids by design and applies the tabletop penalty (wealth revelation and auction restart) rather than silently correcting, preserving this as an observable strategic error. When multiple bidders submit equal highest bids in the same call round, ties are broken by a per-auction random priority order, sampled once when the auction starts. Auction settlement uses a dynamic-programming subset-sum routine to construct the minimum-overpay payment from the winner's money cards (no change is given); Kuhhandel offers and counters, by contrast, are composed card-by-card by the agent, so bluff construction (including 0-value cards) is a deliberate choice rather than an engine heuristic. Full architecture details are in Appendix C, and prompts are in Appendix D.

3.3 SCALABILITY AND CODE AGENTS

The benchmark supports parameterized configurations: set size $k \in \{2, 3, 4, 5\}$ (cards per animal type), animal types (1–10), player count (2–5), and starting hands (0–k). These combine into presets from micro-duel (2 players, k=2, 3 types, $\sim$20 turns, $\sim$50 LLM calls) to marathon (5 players, k=5, 9 types). Our experiments use the standard configuration (4 players, k=4, 10 types).

We provide three deterministic code agents as calibration baselines, each implementing a distinct strategic heuristic with no learning:

1. TRACKERAGENT maintains perfect information from all observable events (tracking revealed cards, opponent holdings, and estimated wealth) and makes greedy decisions conditioned on this state. It bids just above an opponent's estimated budget, exercises buy-right only for quartet-completing cards, and adjusts Kuhhandel offers to the target's inferred cash position.
2. SETRACEAGENT greedily pursues quartet completion, bidding aggressively on near-complete sets and ignoring animals with no path to a quartet. It evaluates each auction card by marginal gain toward a quartet and prioritizes Kuhhandel trades that complete or advance sets, regardless of cost.
3. ECONOMYAGENT models the game as a resource economy, tracking money flows and donkey payouts to exploit cash-poor opponents and avoid enriching cash-rich ones. It caps spending at a fraction of its total wealth per turn, avoids buy-right when the auctioneer surplus would benefit a leading opponent, and bluffs only against opponents with low counter probability.

4 EXPERIMENTS

4.1 SETUP

We evaluate seven cost-efficient LLMs spanning four providers: Claude Haiku 4.5 (HAIKU, Anthropic), Claude Sonnet 4.5 (SONNET, Anthropic), Gemini 2.5 Flash Lite (G2.5-FL, Google), Gem-

Figure 2: **Tournament policy dynamics** (88 canonical games: 60 pure-LLM + 28 mixed comp1). Top row: TrueSkill conservative estimate, win rate, and wealth trajectory (mean coins held per turn). Middle row: accept rate vs. offer size, bluff ratio by donkey-phase (0–4 donkeys on the table), and psychological exploitation metrics. Bottom row: Kuhhandel initiation and bid aggressiveness per phase, and challenger/target win rates.

ini 3 Flash Preview (G3-F, Google), Gemini 3.1 Flash Lite Preview (G3.1-FL, Google), GPT-5.4 Nano (GPT5.4-N, OpenAI), and DeepSeek v3.2 (DS-v3.2, DeepSeek). We pit each LLM against three deterministic code agents (TRACKERAGENT, SETRACEAGENT, ECONOMYAGENT) to calibrate LLM play against algorithmic baselines (Section 4.3).

We run 232 games across two formats: (i) 60 pure-LLM tournaments in 15 balanced four-player combinations, covering every pair of the six original LLMs at least twice; and (ii) 172 mixed-format games where a single LLM competes against three code agents across seven opponent-composition schedules (e.g., Tracker+Economy+SetRace, denoted C1, or 2×Tracker+Economy, denoted C2). All games use canonical auction mode (simultaneous-intent bidding), full deck (10 animal types), full memory mode (scratchpad), temperature 0.1, and the `standard` character prompt with reasoning effort set to low and maximum 4,096 response tokens.

4.2 MAIN RESULTS

Figure 2 summarises the two-format evaluation. On the combined-comp1 slice (60 pure-LLM + 28 mixed-comp1 = 88 canonical games), Gemini 3 Flash ranks first (TrueSkill $\mu_c = 26.2$, 72.7% wins, median score 5,250), followed closely by TrackerAgent ($\mu_c = 24.7$, 53.6%), Gemini 3.1 FL ($\mu_c = 24.3$, 43.2%), and SetRaceAgent ($\mu_c = 23.5$). Two code agents thus outperform four of seven LLMs. The weaker tier spans DS-v3.2 ($\mu_c = 20.7$), Sonnet 4.5 ($\mu_c = 20.0$, n=4), GPT5.4-N ($\mu_c = 19.6$), Haiku ($\mu_c = 18.4$), EconomyAgent ($\mu_c = 18.1$), and G2.5-FL ($\mu_c = 17.3$). The score gap tracks quartet completion: G3-F averages 3.98 quartets per game, while Haiku and EconomyAgent average below 1.7.

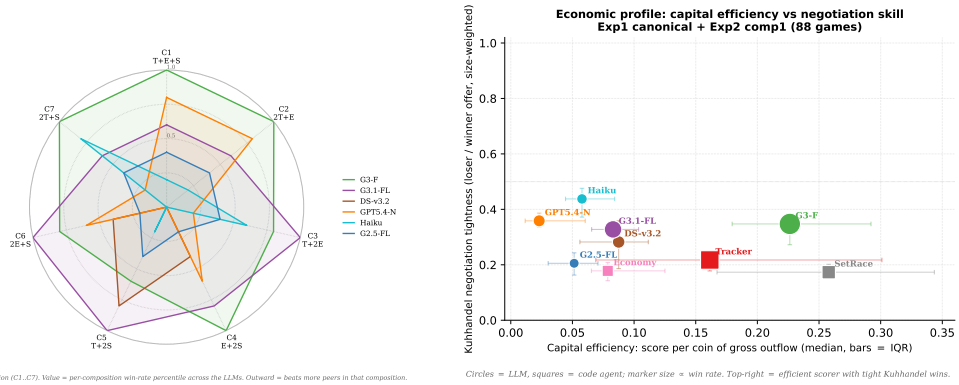


Figure 3: **Left: LLM robustness across 7 code-agent compositions** (exp2_all7, 172 mixed games). Axes C1–C7 are compositions; values are per-composition win-rate percentiles across the LLMs. Outward = beats more peers on that mix; large convex polygon = robust across mixes, spiky polygon = composition-sensitive. **Right: Economic profile** (88 canonical games; Sonnet omitted, n=4). x: capital efficiency η, y: Kuhhandel tightness τ (definitions in Appendix B.1). Median across games, IQR as bars; circles = LLM, squares = code agent; marker size $\propto$ win rate.

4.3 LLM vs. Code-Agent Calibration

The mixed-format games let us separate LLM skill from LLM-vs-LLM variance. In the 172-game exp2 slice (Figure 3a), G3-F wins 67.9% of its 28 games averaged over the seven compositions, and G3.1-FL wins 50.0%. Both clear all three heuristic baselines. The remaining LLMs fall below every code agent by mean score: DS-v3.2 (10.7% wins), GPT5.4-N (14.3%), Haiku (7.1%), and G2.5-FL (0.0%). The hardest compositions for LLMs are those containing two TrackerAgents (C2, C7) — the card-counting pressure compounds and only G3-F still wins a majority. Conversely, compositions with two EconomyAgents (C3, C6) are the softest: every LLM except G2.5-FL wins a larger share there, confirming that EconomyAgent's conservative budgeting is the weakest of the three code heuristics.

TrackerAgent ($\mu_c = 28.5$ in the 172-game slice) exploits observed events via card counting, a capability no LLM replicates despite receiving the same observable information in its prompt. SetRaceAgent ($\mu_c = 27.4$) targets quartet completion aggressively, and EconomyAgent ($\mu_c = 24.1$) relies on conservative resource management. The ordering tracker $>$ race $>$ economy suggests that information exploitation and decisive acquisition matter more than caution. The economic profile in Figure 3b complements this on two rates (full definitions in Appendix B.1). The x-axis is capital efficiency η: score returned per coin of gross outflow; η=0 = coins spent without scoring, η=1 would be one point per coin spent (no agent approaches this; top LLMs sit near 0.25–0.30). Further right = each coin bought more final score. The y-axis is Kuhhandel negotiation tightness τ, the size-weighted $\max(\text{loser}, 10)/\text{winner}$ across counter-wins; τ=1 = the winner paid only marginally above the opponent's counter (maximally tight), $\tau \to 0$ = the winner crushed a small counter with a huge offer (loose, most coins wasted). Further up = fewer coins burned per Kuhhandel won. The ideal corner is top-right: efficient *and* tight. G3-F sits closest, combining the highest η among LLMs with solid $\tau \approx 0.35$. Haiku and GPT5.4-N are the tightest negotiators ($\tau \approx 0.4$) but pay for it with low efficiency. Tracker and SetRace land opposite: high efficiency but loose counters ($\tau \approx 0.2$), trading negotiation precision for acquisition pressure.

4.4 Strategic Behavior

Three mechanisms separate strong play from weak (detailed metrics in Table 3, Appendix B).

Spending efficiency. G3-F converts coins to points at 2.94 points per coin and G3.1-FL at 2.70; TrackerAgent leads at 3.48. G2.5-FL spends aggressively (bid aggressiveness 2.52) but only extracts 0.61 points per coin, overpaying for cards it fails to assemble into quartets. High volume without targeting yields worse outcomes than moderate volume with better targeting. Cost per quar-

tet (Appendix Figure 8) traces the same ordering: code agents and top LLMs secure high-value animals for 400–540 coins per quartet; G2.5-FL pays 1,039.

Resource discipline and self-bidding. The three code agents never overbid. G2.5-FL overbids at 1.20%, the highest rate, and finishes last; Haiku (0.87%), DS-v3.2 (0.49%), and GPT5.4-N (0.47%) also overbid at meaningful rates. Among LLMs, self-bidding correlates inversely with performance: G3.1-FL and Sonnet self-bid in under 7% of rounds, while DS-v3.2, GPT5.4-N, and G2.5-FL self-bid in over 74% of rounds, revealing failure to detect non-competitive contexts. In one game, DS-v3.2 incremented its own bid from 10 to 850 over 49 call rounds as the sole bidder.

Buy-right and phase adaptation. G3.1-FL and G3-F exercise buy-right most frequently among LLMs (31.3% and 26.5%); TrackerAgent tops all agents at 34.4%. G3-F escalates bid intensity from early-game (0.26) to late game (2.49), an $\approx 10\times$ ramp timed to quartet-completion pressure. Sonnet shows an even steeper ramp ($0.14 \rightarrow 2.45$), but with only four games the estimate is noisy. G3.1-FL shows a smaller, coherent ramp ($0.22 \rightarrow 0.65$). G2.5-FL stays at bid aggressiveness 2.07 early *and* 2.08 late — no adaptation, skipping the accumulation phase entirely. DS-v3.2 and GPT5.4-N escalate to similar absolute values (1.65 and 2.09) but without the spending discipline to convert aggression into quartets. The code agents themselves ramp sharply (Tracker $0.06 \rightarrow 1.92$, SetRace $0.11 \rightarrow 1.55$), so endgame escalation is a *rational* pattern the top LLMs have learned and the weakest ones have not.

4.5 Trajectory-Level Observations

Aggregate metrics mask qualitative failures visible in individual traces. We describe three patterns that recur across games and that the code agents never produce.

Bankrupt initiation. G2.5-FL repeatedly initiates Kuhhandel after depleting its money through overbidding. In one game it challenges an opponent for a goose with zero money cards, producing a face-down offer of 0 cards. Because the opponent sees the card count, a 0-card offer has no bluff potential; the opponent counters with the minimum (10 coins) and wins for free. The model fails to condition action choice on resource state.

Adaptive offer calibration. G3-F conditions Kuhhandel offers on both opponent wealth and game context: against bankrupt opponents it deploys 0-value bluffs at high frequency, and against well-funded opponents contesting high-value animals it commits 500+ coins. One trace shows G3-F completing its fourth quartet by paying far above face value for a single card — the multiplicative-scoring formula turned a nominally wasteful overpay into a net gain of $\approx 1{,}800$ points. Weaker LLMs do not exhibit this kind of portfolio-aware price discrimination.

Auction self-escalation. In canonical auctions, DS-v3.2 occasionally treats its own most-recent bid as a competitor's and raises again, incrementing from 10 to 850 over 49 call rounds as the sole bidder. This is reflected in its 75.4% self-bidding rate (Table 3). The code agents never exhibit this failure mode because their bid-choice rule explicitly conditions on the other players' last action.

4.6 Computational Cost and Format Compliance

Token usage varies roughly $20\times$ across models: G3-F emits around 1,500 completion tokens per turn on average (reflecting extended internal reasoning), G3.1-FL about 80. Both rank in the top two, so verbose reasoning chains are not required for strong play. Haiku emits the most completion tokens but ranks eighth — token volume alone does not predict strategic quality. Structured-output failures (invalid JSON, missing fields, truncation) are below 1% for every model we evaluated; format errors trigger a single retry with explicit error feedback and do not explain the performance gaps we observe (Appendix B).

5 Discussion

Strategic coherence—spending efficiency, resource discipline, and adaptive phase play combined—predicts success better than any single metric we measure. The top two LLMs share low self-bidding rates and near-zero overbid rates, indicating correct auction-state modelling. The weakest models exhibit distinct failure modes: G2.5-FL overbids and initiates bankrupt trades; DS-v3.2 self-bids in

over 75% of rounds; GPT5.4-N escalates bids without the spending efficiency to convert; and Haiku participates too passively. None of these patterns appears in the code agents.

TrackerAgent's third-place ranking ($\mu_c = 24.7$, above four of seven LLMs) calibrates the benchmark: card-counting heuristics suffice to outperform most LLMs tested. The code-agent ordering (TrackerAgent $>$ SetRaceAgent $>$ EconomyAgent) shows that information exploitation matters more than greedy quartet-chasing, which in turn outperforms conservative budgeting. This mirrors the LLM ranking, where the top models combine active acquisition with resource discipline, while weaker models either spend recklessly or participate too passively.

Our ranking (G3-F $\gg$ G3.1-FL $>$ DS-v3.2 $\approx$ Sonnet $>$ GPT5.4-N $>$ Haiku $\gg$ G2.5-FL) roughly matches Chatbot Arena Elo (Zheng et al., 2024). It diverges from static benchmarks: Haiku 4.5 achieves the highest SWE-bench Verified score among these models (73.3%) yet ranks eighth in CATTLE TRADE, while G3.1-FL scores below DS-v3.2 and GPT5.4-N on MMLU-Pro yet places second here. Multi-turn strategic play depends on capabilities (state tracking, adaptive resource allocation, structured-output reliability) that static benchmarks do not measure but that conversational evaluations partially capture.

The benchmark's diagnostic value lies in identifying why a model loses, not just that it loses. Overbid frequency, self-bidding rate, bankrupt-initiation patterns, and context-dependent offer calibration are failure modes invisible to both static evaluations and aggregate rankings like Elo. The structured game logs make these behaviours directly observable and quantifiable.

Several limitations apply. All seven LLMs are cost-efficient models run with reasoning effort set to low and a 4,096-token budget; stronger models or higher budgets may avoid the failures we document. We test only full-memory mode and do not ablate memory or temperature. Game counts per agent range from 4 (Sonnet, comp1 only) to 172 (code agents), and TrueSkill estimates for low-count agents carry wider uncertainty. The behavioural metrics are point estimates without confidence intervals. Whether the failure modes generalise to other multi-agent economic settings remains untested.

Some failures may reflect prompt design rather than model limitations. All models receive identical prompts describing game rules and legal actions but we do not explicitly instruct them to track wealth before bidding or check whether other players are bidding before raising. The code agents receive no natural-language prompts at all and still avoid these errors through conditional logic, suggesting the underlying issue is one of reasoning rather than instruction-following. Code agents also operate on structured data with exact arithmetic, while LLMs must parse natural-language observations and track state across turns; some failures classified as reasoning errors (e.g. overbidding) may partly reflect numerical parsing or working-memory limitations.

6 CONCLUSION

We introduced CATTLE TRADE, a multi-agent benchmark integrating auctions, hidden-information bargaining, bluffing, and resource management into one evaluation environment. Across 232 games with seven cost-efficient LLMs and three deterministic code agents, we find that spending efficiency and resource discipline predict success, while overbidding, self-bidding spirals, and undisciplined bluffing characterise failure. Two heuristic code agents each outperform four of seven LLMs, indicating that agent failures arise from limitations in state tracking and decision-making, not only from game complexity.

The behavioural profiles produced by the benchmark diagnose why models fail in ways that aggregate win rates cannot. Overbidding points to weak resource-constraint tracking; self-bidding points to weak auction-state modelling. The code agents serve as interpretable baselines clarifying which strategic capabilities LLMs lack. Gemini 3.1 Flash Lite achieves a top-tier ranking with about 80 completion tokens per turn on average versus about 1,500 for Gemini 3 Flash, showing that verbose reasoning chains are not necessary for competitive play. A natural next step is to run a larger study that includes frontier LLM and enables chat mode. Another future direction is to train open-weight models via self-play in CATTLE TRADE and measure whether self-bidding and overbidding rates decrease. Benchmarking against human players and integrating human data into the training procedure is an interesting future avenue.

REFERENCES

Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, et al. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022. 2

Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890, 2019. 2

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 1, 2

Anthony Costarelli, Mat Hilfiker, Ilya Peshkov, Banghua Zhu Ayyoob, et al. GameBench: Evaluating strategic reasoning abilities of LLM agents. *arXiv preprint arXiv:2406.06613*, 2024. 2

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stinis, and Dimitrios Stamoulis. GTBench: Uncovering the strategic reasoning limitations of LLMs via game-theoretic evaluations. *arXiv preprint arXiv:2402.12348*, 2024. 1, 2

Kanishk Gandhi, Dorsa Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Raghunathan, and Noah D Goodman. Strategic reasoning with language models. *arXiv preprint arXiv:2305.19165*, 2023. 2

Jiaxian Guo, Bo Yang, Paul Yoo, Bill Yuchen Lin, Yusuke Iwasawa, and Yutaka Matsuo. Suspicion-agent: Playing imperfect information games with theory of mind aware GPT-4. *arXiv preprint arXiv:2309.17277*, 2024. 2

Thilo Hagendorff. Deception abilities emerge in large language models. *Proceedings of the National Academy of Sciences*, 121(24):e2317967121, 2024. 2

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. 1, 2

Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill: A Bayesian skill rating system. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. 4, 11

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. 1

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-bench: How good are LLMs at playing games? In *International Conference on Learning Representations*, 2026. 2

Jingru Jia, Zehua Yuan, Junhao Pan, Paul E. McNamara, and Deming Chen. LLM strategic reasoning: Agentic study through behavioral game theory. In *Advances in Neural Information Processing Systems*, volume 38, 2025. 2

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2443–2453, 2017. 1, 2

Jonathan Light, Min Cai, Sheng Xu, and Ziniu Huang. From text to tactic: Evaluating LLMs playing the game of Avalon. *arXiv preprint arXiv:2310.05036*, 2023. 1, 2

Wenye Lin, Jonathan Roberts, Yunhan Yang, Samuel Albanie, Zongqing Lu, and Kai Han. GAME-BoT: Transparent assessment of LLM reasoning in games. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 7656–7682, 2025. 2

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. 2

Shaoguang Mao, Yuzhe Cai, Yan Xia, Wenshan Wu, Xun Wang, Fengyi Wang, Qiang Guan, Tao Ge, and Furu Wei. ALYMPICS: LLM agents meet game theory. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pp. 2845–2866, 2025. 2

Alexander Pan, Jun Shern Chan, Andy Zou, Nathaniel Li, Steven Basart, Thomas Woodside, Hanlin Zhang, Scott Emmons, and Dan Hendrycks. MACHIAVELLI: A benchmark for understanding agentic AI harms. In *International Conference on Machine Learning*, 2024. 1, 2

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. 2

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. Exploring large language models for communication games: An empirical study on Werewolf. *arXiv preprint arXiv:2309.04658*, 2023. 1, 2

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2024. 8

A Game Rules Summary

Table 2 lists all animal types and their quartet values. The game uses 10 animal types with 4 cards each (40 cards total).

Table 2: Animal types and quartet values in *Kuhhandel*.

Animal	Quartet Value (points)
Chicken	10
Goose	40
Cat	90
Dog	160
Sheep	250
Goat	350
Donkey	500
Pig	650
Cow	800
Horse	1,000

Auction flow. The active player draws a card and announces an auction. Other players submit bids simultaneously (canonical mode). The auctioneer sees the highest bid and decides: accept (card goes to bidder, money to auctioneer) or buy-right (auctioneer pays the bid amount to the bidder, keeps the card). If no bids are placed, the auctioneer receives the card for free. If a player bids more than they have, their wealth is revealed to all players, the auction is repeated, and the penalized player is limited to bidding their true valid amount.

Kuhhandel flow. The active player selects an opponent who shares at least one animal type. The initiator places money cards face-down as an offer. The opponent chooses: (a) *accept* take the money (without seeing it) and give all matching animals; or (b) *counter* place their own face-down offer. If countered, both offers are revealed and exchanged: each player receives the other's offer. The player who *offered more money* wins and takes the animals (paying more to "buy" them). Bluffing (0 offers) only works when the opponent *accepts* without seeing; if they counter with any positive amount, they beat the bluff. The initiator's advantage is choosing when to attack and winning after 3 ties. On ties (equal amounts), the trade repeats; after 3 ties, the initiator wins by default.

Donkey payout. When a donkey card is drawn for auction, all players immediately receive a money card from the bank (50, 100, 200, or 500 depending on which donkey card). This injects liquidity into the game economy, changing bidding dynamics.

B Detailed Results

B.1 Metric Definitions

All per-agent metrics referenced in §4 and used in the figures and tables of this appendix are computed from raw game logs as follows. $\mathcal{Q}$ denotes the set of completed quartets and v_q the quartet value (§3.1).

Final score S**.** $S = \big(\sum_{q\in\mathcal{Q}} v_q\big) \cdot |\mathcal{Q}|$ (Eq. 1); evaluated at end of game.

Capital efficiency η **(≡ *Spend Eff.* in Table 3).** $\eta = S/\mathrm{out}$ per game, where out is gross outflow: the total coins leaving the agent's hand, summed across auction payments (when winning a bid), Kuhhandel offers paid in accepted trades and counter-exchanges, and buy-right payments. Reported as median across games.

Kuhhandel negotiation tightness τ**.** For each counter-exchange won by the agent in a game, let w_k be the winner's offer and ℓ_k the loser's. Then $\tau = \min\big(\sum_k \max(\ell_k, 10) \,/\, \sum_k w_k,\ 1\big)$, size-weighted per game (summing over counter-wins k). The 10-coin floor on ℓ_k avoids division artefacts when defeating a 0-bluff. $\tau{=}1$ = winner paid only marginally above the opponent's counter; $\tau{\to}0$ = winner crushed a small counter with a much larger offer.

Bid aggressiveness. Mean of bid$/v_q$ over all auctions in which the agent bid, where v_q is the quartet value of the auctioned animal.

Buy-right %. Fraction of auctioneer decisions in which the agent exercised buy-right (keeping the card) rather than selling to the highest bidder.

Kuhhandel-accept %. Fraction of Kuhhandel challenges (agent as target) resolved by accepting the face-down offer rather than countering.

Bluff %. Fraction of the agent's Kuhhandel offers consisting purely of 0-value money cards.

Self-bid %. Fraction of the agent's auction bids placed in rounds where the agent is already the current high bidder (i.e., raising against itself).

Overbid %. Fraction of auctions in which the agent submitted a bid exceeding its total money; triggers the canonical-mode wealth-revelation penalty (§3.2).

Win rate. Fraction of games in which the agent achieved the highest final score S.

TrueSkill μ, σ**.** Bayesian skill estimate (Herbrich et al., 2006) fit across all games in the slice; $\mu_c = \mu - 3\sigma$ is the conservative estimate used in rankings.

Agent	Bid Agg.	Buy-Right	KH-Accept	Bluff%	Self-Bid	Spend Eff.	Overbid%
Gemini 3 Flash	0.94	26.5%	36.3%	8.8%	21.7%	2.94	0.00%
TrackerAgent	1.14	34.4%	0.0%	27.0%	58.7%	3.48	0.00%
Gemini 3.1 Flash Lite	0.36	31.3%	32.1%	7.0%	6.8%	2.70	0.35%
SetRaceAgent	0.83	31.2%	34.8%	23.0%	67.0%	2.65	0.00%
DeepSeek v3.2	0.82	18.2%	45.1%	17.0%	75.4%	1.33	0.49%
Claude Sonnet 4.5	0.76	23.1%	38.1%	27.8%	1.8%	2.18	0.00%
GPT-5.4 Nano	1.07	27.6%	28.6%	15.2%	74.6%	1.80	0.47%
Claude Haiku 4.5	0.31	20.0%	26.4%	15.6%	9.1%	2.37	0.87%
EconomyAgent	0.67	14.5%	14.8%	30.6%	59.6%	2.27	0.00%
Gemini 2.5 Flash Lite	2.52	15.9%	51.1%	25.7%	78.5%	0.61	1.20%

Table 3: Strategic-behaviour profiles across 88 canonical games (combined-comp1 slice). Bid Agg.: bid-to-value ratio. Buy-Right: fraction of auctioneer decisions keeping the card. KH-Accept: Kuhhandel challenges accepted. Bluff%: pure 0-value offers. Self-Bid: bids in rounds with no competing bid since this agent's last bid. Spend Eff.: points per coin spent. Overbid%: overbid frequency.

Score variance is high: G3-F ranges from 0 to over 10,000 (mean 5,936, std 4,091), reflecting the multiplicative scoring formula's sensitivity to quartet count. G2.5-FL shows the highest relative variance (std 1,292 on mean 1,115).

Figure 4 provides detailed auction and Kuhhandel metrics across the ten agents. The cross-play win-rate heatmap (bottom panel) shows G3-F winning the majority of games against every opponent, including all three code agents. G3.1-FL maintains positive records against all models except G3-F. TrackerAgent outperforms four of seven LLMs, while DS-v3.2 only beats weaker LLMs.

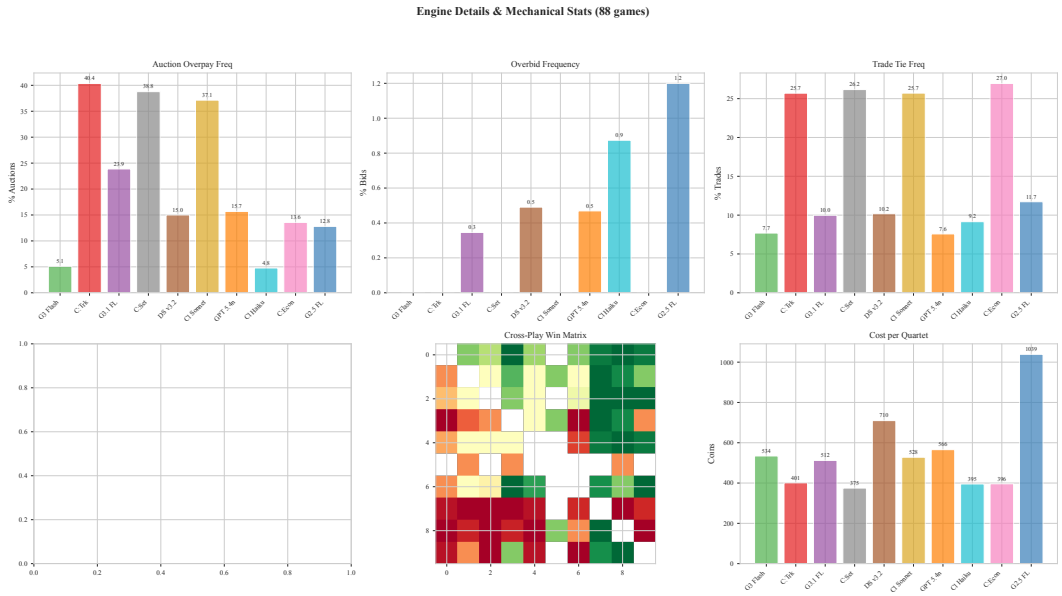


Figure 4: Engine-level metrics (88 canonical games, 7 LLMs + 3 code agents): auction participation, win rates, overpay/overbid frequencies, Kuhhandel counter and challenger/target win rates, bluff magnitude, tie frequency, token usage, truncation, and cost per quartet.

Figure 5 shows positional analysis, economic trajectories, and phase-dependent behaviour. The positional win-rate heatmap (left) confirms no systematic seat advantage for strong models. Money trajectories (centre) show G3-F maintaining higher liquidity throughout games, while G2.5-FL's cash reserves deplete rapidly under overbidding penalties. Code agents show stable money trajectories under their deterministic rules. The accept-rate-by-offer-size panel (right) shows acceptance probability rising with the number of cards at stake, suggesting models calibrate defensive decisions to trade magnitude.

Token usage per game varies roughly 20×: G3-F generates about 275,000 completion tokens per game, while G3.1-FL uses about 14,800. Both rank in the top tier, so verbose reasoning chains are not required for strong strategic play in this setting.

C Agent Architecture

Observation space. At each decision point, player i receives an observation o_t^i rendered as natural language. The observation comprises four components:

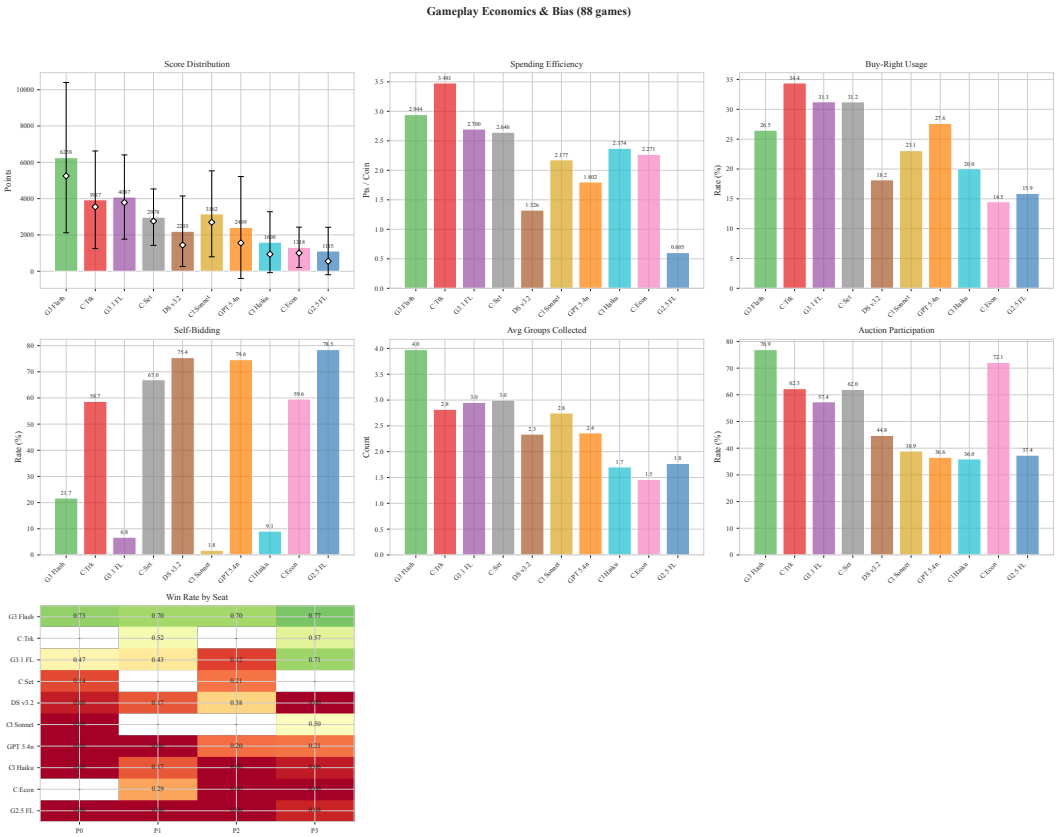


Figure 5: Governance-level analysis (88 canonical games, 7 LLMs + 3 code agents): seat-position win rates, money trajectories, accept rate by offer size, exploitation metrics, bid-to-wealth ratio, and Kuhhandel initiation by donkey phase.

1. **Private state**: exact money card denominations (e.g., 0×2, 10×4, 50×1) and animal holdings with counts.
2. **Public state**: all opponents' animal holdings (type and count), opponents' money card *counts* (not values), number of cards remaining in the deck, and turn number.
3. **Decision context**: parameters specific to the pending decision, including the card being auctioned and current bids (auction), or the animal type, challenger/target identities, and tie count (Kuhhandel).
4. **Memory** (if enabled): the most recent $\sim$10 game events ($\sim$150 tokens) and the agent's scratchpad contents (see below).

Information asymmetry is maintained: money card *values* of other players are hidden; only card counts are visible. A player's exact wealth is revealed only when they overbid in an auction, mirroring the tabletop game's information structure.

Action space. The action space $\mathcal{A}_t$ is context-dependent. The engine presents a decision prompt and the agent responds with a structured JSON object. Seven action types arise:

1. **Turn choice**: {`auction, kuhhandel`}:select whether to draw a card for auction or initiate a Kuhhandel trade with a valid target.
2. **Auction bid**: {`bid(`*amount*`), pass`}:bid on the auctioned card or pass. Bids must be multiples of 10.
3. **Buy-right decision**: {`sell, buy_right`}:as auctioneer, accept the highest bid (sell) or exercise the buy-right by paying the bid amount to the bidder.

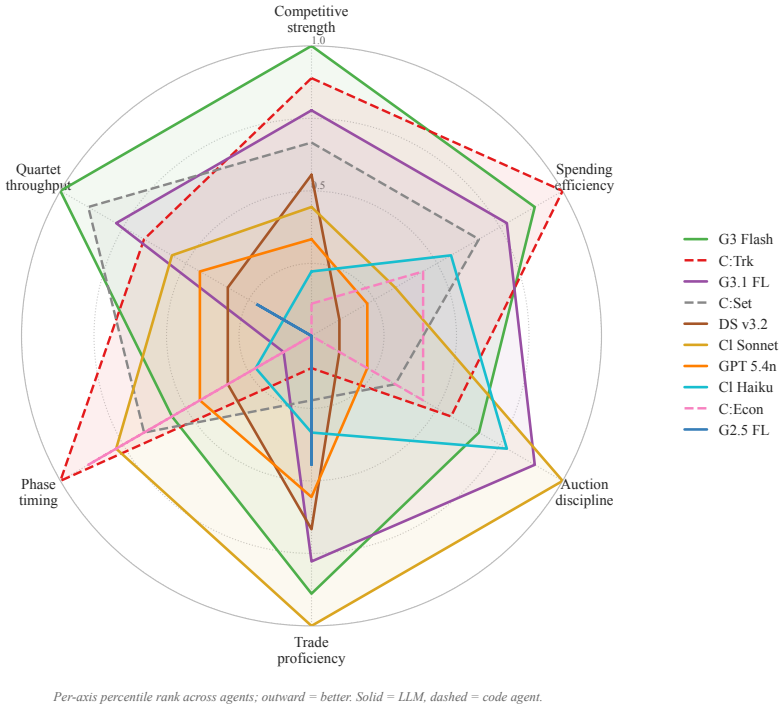


Figure 6: **Strategic-profile radar** across six orthogonal axes (88 canonical games, all 10 agents). Each axis is percentile-normalised across the ten agents, so outward means *better*: competitive strength (TrueSkill μ_c), spending efficiency (points per coin), auction discipline ($1 - \frac{1}{2}$(overbid + self-bid)), trade proficiency (Kuhhandel challenger win rate), phase timing (log late/early bid-aggressiveness ratio), and quartet throughput (mean quartets per game). Solid = LLM, dashed = code agent.

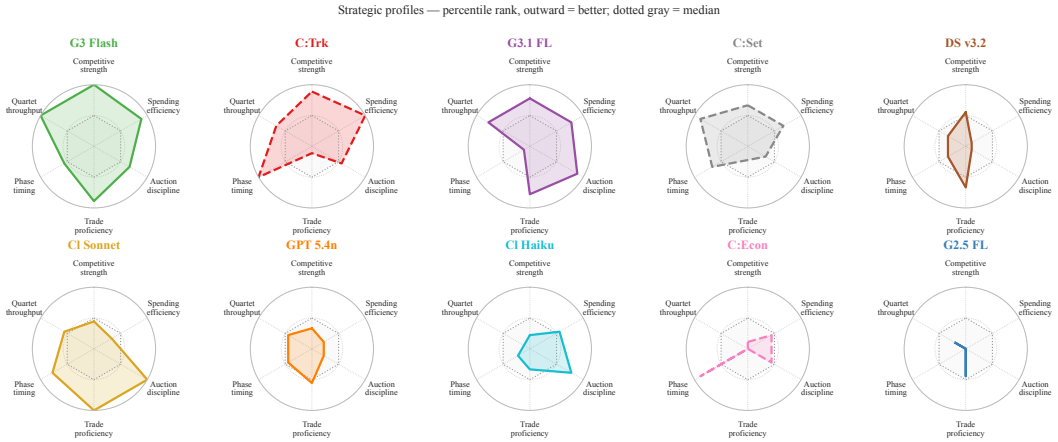


Figure 7: Strategic-profile radar per agent (88 canonical games). Same six axes as Figure 6; each agent plotted against the cross-agent median (dotted grey). The small-multiples view makes per-agent shape asymmetries easier to read than the overlaid version.

4. **Kuhhandel initiation**: {`target, animal, offer_cards`}:select opponent and animal type, then select specific money cards from hand as the face-down offer (including 0-value cards for bluffing).

5. **Kuhhandel defense**: {`accept, counter(cards)`}:accept the face-down offer (take money sight-unseen, surrender animals) or counter with specific money cards from hand.

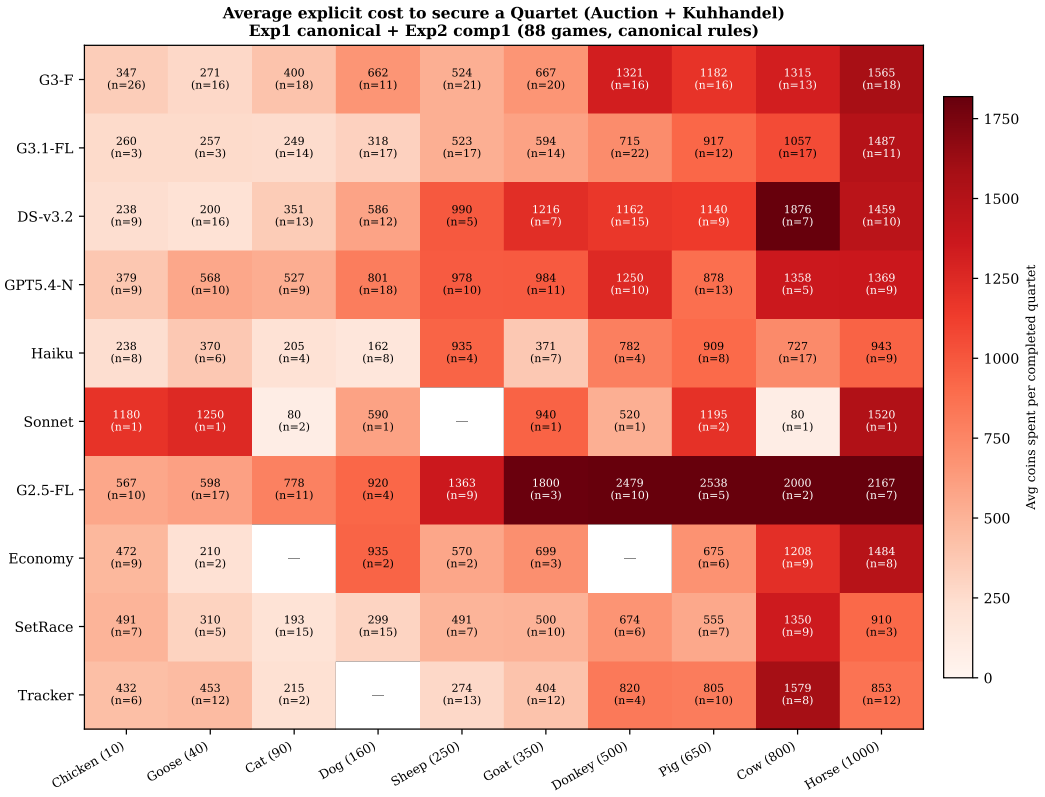


Figure 8: **Gross-outflow cost per quartet** per animal (88 canonical games). Rows: 7 LLMs then 3 code agents; columns: 10 animal types in ascending value order. Cell value: total coins spent by that agent on that animal divided by quartets completed; n: quartets completed. This is a first-order view; it does not account for Kuhhandel inflows or the multiplicative scoring effect (see the net-ledger analysis in Figure 3b).

6. **Kuhhandel tie retry**: {`offer_cards`}:after a tie, submit a new offer using specific cards. After three ties, the initiator wins by default.

7. **Scratchpad update**: {`notes`}:free-text strategic notes (full memory mode only).

Error handling by action type. The framework validates all actions and applies action-specific recovery procedures. Table 4 summarizes the handling for each error case.

Payment resolution. Because no change is given, the engine must select a subset of money cards whose sum meets or exceeds the required amount while minimizing overpayment. This is a variant of subset sum solved via dynamic programming over the player's money cards. The algorithm finds the smallest achievable sum $\geq$ target and, among ties, the combination using the fewest cards. For example, paying 60 with cards $\{100, 50, 10, 10, 0, 0\}$ selects 50+10 (exact) rather than 100 (overpay by 40).

Scratchpad mechanism. In *full* memory mode, each agent maintains a personal scratchpad s^i, a free-form text buffer capped at $\sim$300 tokens. At the end of a player's own turn, if new observable events occurred, the framework issues an additional LLM call providing the current scratchpad contents and the new events, and requests updated notes. The agent uses the *same model* as for game decisions, preserving reasoning style consistency. The updated scratchpad is included in all subsequent observations under a "YOUR NOTES" heading.

Action Type	Error Case	Handling
All actions	Unparseable JSON	Multi-stage fallback: (1) direct parse, (2) regex extraction of last JSON object from free text, (3) manual key-value extraction with type coercion
All actions	Response truncated by token limit	Lightweight retry call requesting only the JSON action (512 tokens, reasoning disabled)
All actions	API failure	Exponential backoff retry (1s, 2s, 4s) up to configurable max attempts
Auction bid	Non-multiple of 10	Rounded down to nearest multiple of 10
Auction bid	Bid below minimum	Clamped up to minimum bid increment
Auction bid (canonical)	Bid exceeds wealth	*Not corrected.* First overbid triggers wealth revelation and auction restart; a second overbid in the same auction eliminates the player
Auction bid (fast)	Bid exceeds wealth	Clamped to total money
Buy-right	Cannot afford payment	Action rejected; card sold to bidder
Kuhhandel initiation	Invalid target (wrong animal, self-trade)	Observation includes pre-computed valid targets; if the LLM names an invalid target or animal, the agent falls back to the first valid (target, animal) pair
Kuhhandel initiation	Offer cards not in hand	One retry with error feedback showing available cards; fallback to greedy card selection
Kuhhandel defense	Counter cards not in hand	Same retry-then-greedy procedure as initiation
Payment (all types)	Exact amount not possible	DP algorithm finds minimum-overpay card combination. No change given, matching tabletop rules

Table 4: Error handling by action type. The framework distinguishes between errors that are silently corrected (bid rounding), errors that trigger retries (malformed JSON, invalid cards), and errors that are game mechanics (overbidding). The canonical auction mode intentionally permits overbids because the resulting wealth revelation is a strategically meaningful penalty.

D AGENT PROMPTS

All prompts below are reproduced verbatim from the benchmark implementation. Template variables (in angle brackets) are filled dynamically from game state. Every action prompt ends with a standardized JSON response instruction.

D.1 SYSTEM PROMPT (GAME RULES)

The rules prompt is included in every LLM call as a system message.

```
KUHHANDEL MASTER - RULES

GOAL: Complete quartets (4 cards of same animal).
Score = (sum of complete quartet values) × (number of quartets).
Example: 1 donkey quartet (500) + 1 horse quartet (1000)
         = (500+1000) × 2 quartets = 3000 points.
Example: 1 horse quartet alone = 1000 × 1 = 1000 points.
         Incomplete sets score 0.

ANIMALS (Complete Quartet Value):
```

```
Chicken=10, Goose=40, Cat=90, Dog=160, Sheep=250,
Goat=350, Donkey=500, Pig=650, Cow=800, Horse=1000

YOUR TURN - Choose one:

[A] AUCTION: Draw a card from the deck. Other players bid
    (each bid must exceed the previous by at least 10 coins).
    Then you (the auctioneer) choose:
    - ACCEPT highest bid: they pay you, get animal
    - BUY-RIGHT: you pay that amount TO highest bidder,
      YOU keep animal
    No change given - must pay exact or overpay.
    OVERBID: If the highest bidder cannot pay, they must
    reveal all their money cards. The auction restarts and
    the overbidder may not overbid again.

[B] KUHHANDEL (Cattle Trade): Trade an animal type you
    BOTH own with an opponent.
    - You place a hidden offer face-down (using coin cards;
      zeros allowed for bluffing)
    - Opponent chooses:
      - ACCEPT: takes your hidden coin cards (without seeing
        them first), gives you their animal
      - COUNTER: places their own face-down offer
    - If countered: heaps are swapped and counted secretly
      by each player. Whoever OFFERED more coins wins and
      takes the animal(s). The exact amounts and card counts
      stay private -- other players learn only who won.
    - Ties: repeat with new offers up to 3x, then challenger
      wins automatically.
    - If both have 2+: ALL cards of that type trade.
```

D.2 Character Prompt

Each agent receives a character prompt prepended to all observations. The `optimal` character used in our experiments:

```
You are an AI playing Kuhhandel Master. Your goal is to win.
Play optimally to maximize your expected score.
```

D.3 Observation Format

At each decision point, the agent receives a structured observation:

```
=== GAME STATE (You are Player <id>) ===
Turn: <turn_number>
Cards remaining in deck: <deck_remaining>

YOUR HAND:
  Money: <total> coins total
  Money cards: 100 coins, 50 coins, 10 coins, 10 coins,
               0 coins, 0 coins
  Animals: 3x Horse, 2x Cow, 1x Sheep

OTHER PLAYERS:
  Player 1: 2x Dog, 1x Sheep | 5 coin cards
  Player 2: 1x Horse, 2x Chicken | 3 coin cards
  Player 3: 2x Cow, 1x Goat | 7 coin cards

VALID KUHHANDEL TARGETS:
  Player 2: horse (you have 3, they have 1)
  Player 3: cow (you have 2, they have 2)

RECENT EVENTS (what you observed):
```

```
  Turn 21: Player 1 won auction for dog (40 coins)
  Turn 22: Kuhhandel: Player 2 challenged Player 3
           for cow. Player 3 countered. Player 2 won.

YOUR NOTES:
  <scratchpad contents, full memory mode only>
```

Opponent money card *counts* are visible but not values, matching the tabletop game's information structure. A player's exact wealth is revealed only through overbid events.

D.4 Turn Choice

When it is the active player's turn, they choose between auction and Kuhhandel:

```
<observation>

It's your turn. You must choose ONE action:
1. AUCTION - Flip a card from the deck and auction it
2. KUHHANDEL - Challenge another player to trade an
   animal you both own

Respond with JSON:
{"reasoning": "...", "action": "auction" or "kuhhandel"}
```

D.5 Canonical Auction Bidding

In canonical mode, all non-auctioneer players submit bids simultaneously each round:

```
<observation>

CANONICAL AUCTION - Simultaneous Bidding Round
Round <n>
- Card: sheep (quartet value: 250)
- Current price: 30
- Current winner: Player 3
- You have: 1 of this animal
- Your money: 170 coins (100, 50, 10, 10, 0, 0)

All bidders submit simultaneously. To win, bid HIGHER
than current price. You may bid more than you can pay.
If you win and cannot pay, your total wealth is revealed
to all players and the auction restarts from 0. A second
overbid eliminates you from this auction.

IMPORTANT: Bids must be multiples of 10 coins.

Respond with JSON:
{"reasoning": "...", "action": "pass" or "bid",
 "amount": <multiple of 10>}
```

The prompt explicitly states overbid consequences. Models that overbid reveal their wealth to all opponents and are excluded from the restarted auction.

D.6 Auctioneer Buy-Right Decision

After bidding concludes, the auctioneer decides whether to sell or exercise buy-right:

```
<observation>

BUY-RIGHT DECISION (you are auctioneer)
- Card: sheep (quartet value: 250)
- Highest bid: 60 by Player 1
```

```
- You have: 1 of this animal
- Your money: 170 coins (100, 50, 10, 10, 0, 0)

Options:
- SELL: Take 60 coins, Player 1 gets the card
- BUY_RIGHT: Pay 60 coins to Player 1, YOU keep the card

Respond with JSON:
{"reasoning": "...", "decision": "sell" or "buy_right"}
```

D.7 KUHHANDEL INITIATION

The active player selects a target and constructs a face-down offer from specific money cards:

```
<observation>

KUHHANDEL - Choose target and make offer:
Valid targets:
- Player 1: horse (you have 2, they have 1)
- Player 3: cow (you have 1, they have 2)

Remember:
- You're trading animals you BOTH own
- Your hidden offer is a set of money cards from your hand
- The opponent sees how many cards you offer, but not
  their values
- You can include 0-coin cards to make your offer look
  bigger (bluff)
- Opponent will either ACCEPT (take money, give animal)
  or COUNTER

Your available card values: [100, 50, 10, 10, 0, 0]
Select exactly which cards to include in your offer.

Respond with JSON:
{"reasoning": "...", "target_player": <id>,
 "animal": "<type>",
 "offer_cards": [10, 10, 0, 0]}
```

The model selects concrete card values from its hand. The framework validates that the requested cards exist; if not, it retries with an error message showing available cards.

D.8 KUHHANDEL DEFENSE

The target of a Kuhhandel chooses to accept the hidden offer or counter:

```
<observation>

KUHHANDEL - You are being challenged!
- Player 1 wants your cow
- You have 2 of this animal (quartet value: 800)
- Trade size: 2 card(s)
- Their offer is face-down (3 card(s), unknown value)

Options:
1. ACCEPT - Take their hidden money, give them your
   animal(s)
2. COUNTER - Select cards from your hand as counter-offer.
   Offers are revealed and exchanged. Winner = whoever
   OFFERED more money. Loser gives up animal(s).

Your available card values: [50, 10, 10, 0, 0]
If countering, select exactly which cards to include.
```

```
Respond with JSON:
{"reasoning": "...", "action": "accept" or "counter",
 "counter_cards": [50, 10, 10]}
```

D.9 KUHHANDEL TIE RETRY

After a tie (equal offer values), both players make new offers:

```
<observation>

KUHHANDEL TIE - Make a new offer!
- Trading: cow with Player 3
- Previous offers tied (your offer was ~60)
- Tie count: 1 (after 3 ties, you win automatically)

Your available card values: [100, 50, 10, 10, 0, 0]
Select exactly which cards to include in your new offer.

Respond with JSON:
{"reasoning": "...",
 "offer_cards": [50, 10, 10, 0]}
```

D.10 SCRATCHPAD UPDATE

In full memory mode, the framework issues an additional call after observable events to update the agent's scratchpad:

```
Update your game notes based on these new events.
Keep under 300 tokens.
Focus on: opponent wealth signals, behavioral patterns,
quartet progress, trade outcomes.

Current notes:
<previous scratchpad or "(none yet)">

New events:
Turn 23: Player 1 won auction for dog (40 coins)
Turn 24: You initiated Kuhhandel with Player 2 for horse.
  You offered 50, they countered 0. You won, got 1 horse.
Turn 25: Player 3 completed quartet (sheep)

Respond with ONLY the updated notes, nothing else.
```

The scratchpad is capped at approximately 300 tokens and uses the same model as game decisions. It is included in all subsequent observations under the "YOUR NOTES" heading.

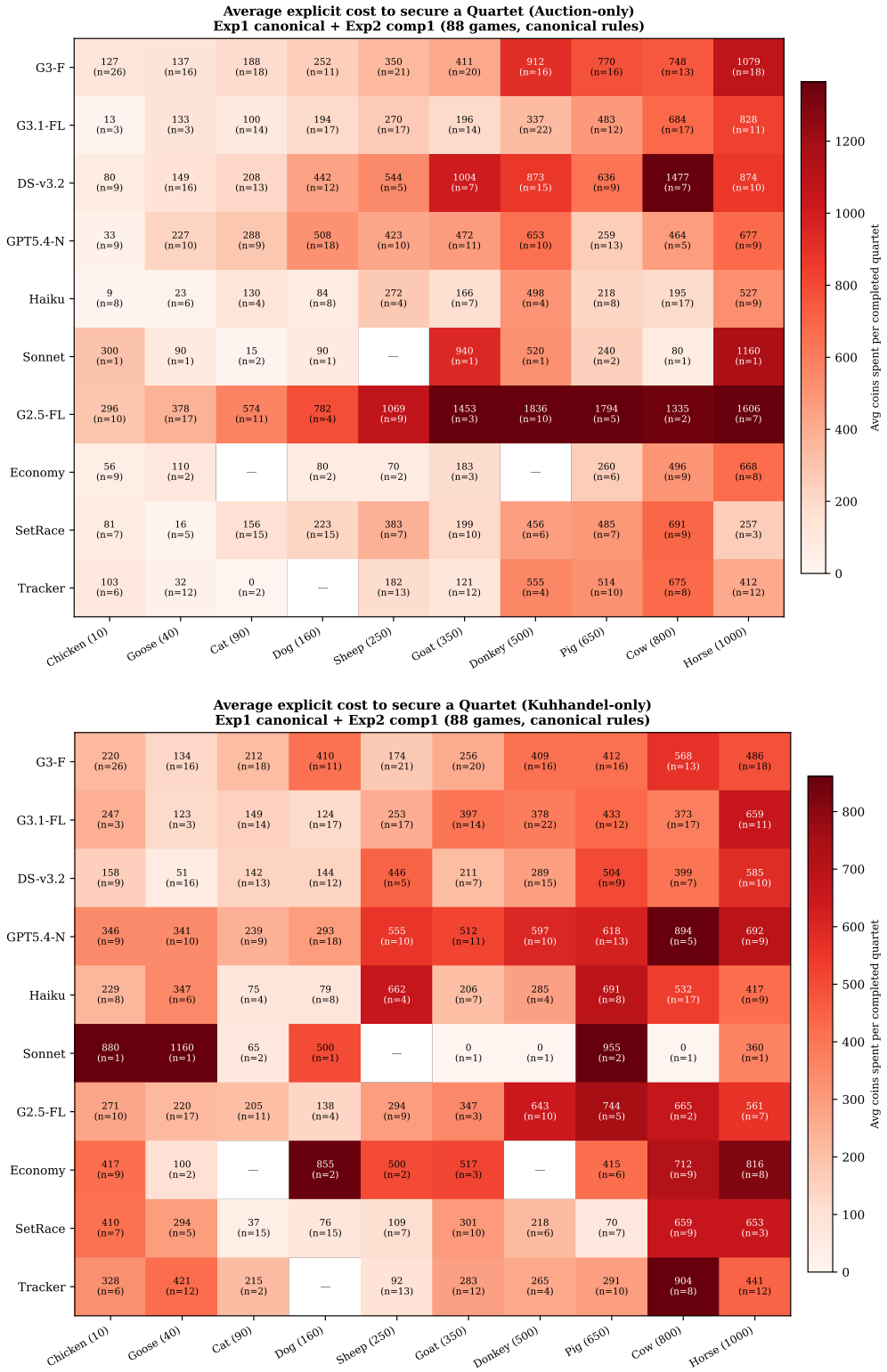


Figure 9: Explicit cost decomposition per animal (88 canonical games). **Top:** auction-only final prices paid by the winner. **Bottom:** Kuhhandel-only winner's last-round offer value. Auction costs dominate for high-value animals; Kuhhandel costs concentrate on mid-tier animals.

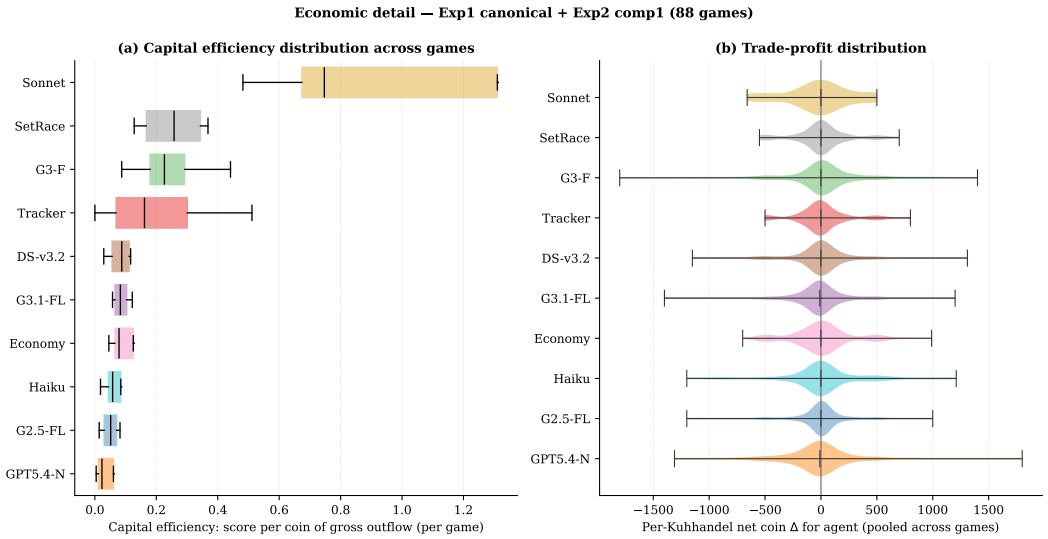


Figure 10: **Economic detail** (88 canonical games). **(a)** Per-game capital efficiency distribution, η = score/gross outflow. Sonnet (n=4) is high-variance and should be read with caution. **(b)** Pooled per-Kuhhandel net coin Δ per agent. Mass right of zero = agent pockets money on average; mass left = agent pays to win animals in Kuhhandels. The main-body scatter (Figure 3b) aggregates these two marginals into per-agent medians with IQR.