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

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents are moving into economic settings such as negotiation, procurement and trading, where success depends less on any single skill than on combining bidding, bargaining, bluffing and budgeting over a long interaction with self-interested opponents. Benchmarks test these skills largely in isolation. We introduce CATTLE TRADE[1], a four-player trading game that requires all of them at once: auctions with a buy-right, face-down trade offers that make bluffing possible, money without change, and a multiplicative score that turns every purchase into a portfolio decision. We evaluate current frontier models, some at two reasoning-effort levels, in a balanced league against deterministic heuristics, and compare them with an earlier generation of cost-efficient models. Frontier models have overtaken heuristics that beat most earlier models, but the field is wide, and neither price nor reasoning budget predicts rank: a low-cost model cannot be separated from the leader, and a frontier model at low effort wins no game. Strong play comes in distinct styles, trading animals away from opponents or buying cheaply and patiently, while weak policies either hoard money or spend it without completing sets. Across all models, play on one's own portfolio is competent, but exploiting opponents is almost absent, and money goes unspent when it decides the endgame. Long multi-agent games thus reveal how models combine skills and model their opponents, which isolated tests do not measure.

1 INTRODUCTION

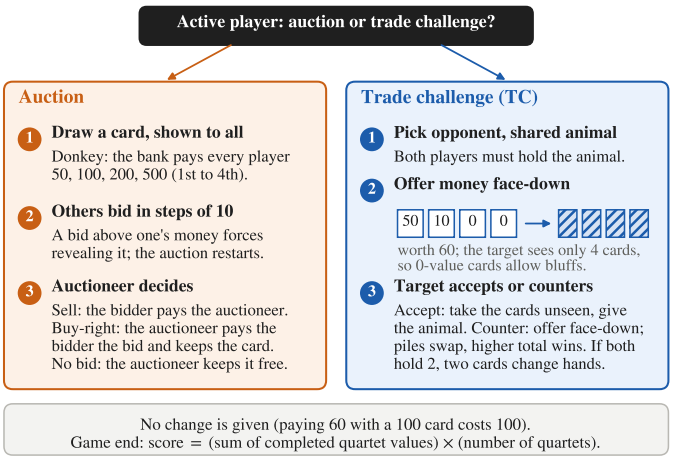
LLM-based agents are increasingly applied to multi-agent settings, from negotiation (Gandhi et al., 2023) to collaborative software development (Hong et al., 2024). Standard benchmarks such as MMLU (Hendrycks et al., 2021) and HumanEval (Chen et al., 2021) measure knowledge and code generation but tell 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., 2023) each test individual competencies in isolation rather than requiring their combination within a single environment.

Passing individual tests does not guarantee joint competence. One key challenge in evaluating agents is not whether models possess isolated skills, but whether they can integrate them coherently under uncertainty, competition, and resource allocation constraints. This matters for downstream agentic settings such as negotiation, procurement, portfolio management and other economic interactions, where effective behavior depends on combining tasks like bidding, bargaining, bluffing, and resource management over time. Prior benchmarks often probe subsets of these abilities, but rarely test whether agents can deploy them jointly in a strategically structured adversarial environment.

We introduce CATTLE TRADE, a benchmark that integrates (i) competitive auctions with a buy-right mechanism coupling bidding strategy to future liquidity, (ii) hidden-offer bilateral trade challenges (TCs) 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 eight LLMs from five providers and three deterministic code agents in a 182-game league in which every pair of the 14 policies shares

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

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.



	Policy	Eff.	Margin	TS μ	Win%
1	Fable 5.1	H	0.68	30.6	60
2	Gemini 3.6 Fl.	L	0.55	29.5	46
3	Opus 5.5	H	0.55	29.1	50
4	Opus 5.5	L	0.42	28.9	40
5	Fable 5.1	L	0.32	27.6	35
6	Gemini 3.8 Fl.	L	0.30	27.7	31
7	GPT-6 Sol	L	0.09	26.1	21
8	DeepSeek 4.1 Fl.	L	−0.08	24.6	19
9	*TrackerAgent*		−0.27	23.1	19
10	Sonnet 5	H	−0.27	23.2	8
11	GLM-5.3	L	−0.32	22.7	8
12	*SetRaceAgent*		−0.52	21.7	8
13	*EconomyAgent*		−0.70	19.7	6
14	Sonnet 5	L	−0.75	19.4	0

Figure 1: **(a) Rules of a turn.** The active player either auctions a card or starts a trade challenge with a face-down money offer; hidden offer values make 0-value bluffs possible. **(b) League ranking** of the 14 policies over 182 games, sorted by margin score (Section 4.1). Eff.: reasoning effort (H high, L low); Fl.: Flash; shaded rows: deterministic code agents. Win%: share of games won (25% for four equal players).

a table 12 times. Three frontier models play at two reasoning-effort levels, which isolates the effect of the configured effort setting on play. We compare this frontier tier with an earlier tier of seven cost-efficient LLMs (242 games) through the code agents that both tiers share.

Frontier models have overtaken the heuristics that beat most cost-efficient models, but they differ widely, and neither price nor reasoning budget predicts their rank. Strong policies win in different ways, some by trading animals away from opponents and others by buying cheaply and patiently, while weak policies either hoard money or spend it without completing sets. Almost no model exploits its opponents: all play their own portfolio competently, but they rarely raise the price of a card another player needs and do not use what they know about an opponent's money.

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 from a balanced 182-game league of 11 LLM policies and three code agents, including a paired high-versus-low reasoning-effort ablation, showing a ranking that is stable across the two halves of the schedule, most LLM policies outperforming the code-agent baselines, an effect of reasoning effort that depends on the model, and a comparison with an earlier tier of cost-efficient models through the shared code agents.

2 Related Work

MMLU (Hendrycks et al., 2021), BIG-bench (Srivastava et al., 2023), and HumanEval (Chen et al., 2021) measure knowledge, reasoning, and code generation. AgentBench (Liu et al., 2024) includes interactive environments and a two-player card game. Cattle Trade instead evaluates auctions, bargaining, and bluffing together under hidden information and resource constraints.

GTBench (Duan et al., 2024) evaluates short game-theoretic interactions. AvalonBench (Light et al., 2023) and Werewolf (Xu et al., 2023) test social deduction. MACHIAVELLI (Pan et al., 2023) uses long-horizon, single-agent text games. Suspicion-Agent (Guo et al., 2024) studies imperfect-information card games; GameBench (Costarelli et al., 2024) covers multiple games; and lmgame-Bench (Hu et al., 2026) provides a Gym-style interface with perception and memory scaffolds. Each of these benchmarks emphasizes a subset of strategic skills. To our knowledge, no existing

benchmark requires agents to combine auctions, hidden-offer trade challenges, bluffing, and discrete resource management within a single game.

Jia et al. (2025) study strategic reasoning through behavioral game theory. GAMEBoT (Lin et al., 2025) decomposes game reasoning into modular problems, while ALYMPICS (Mao et al., 2025) includes a multi-round auction for scarce resources. Pluribus (Brown & Sandholm, 2019) uses search and self-play for poker. CICERO (Bakhtin et al., 2022) combines a dialogue language model with planning for Diplomacy. Negotiation studies include neural dialogue agents (Lewis et al., 2017) and language-model prompting (Gandhi et al., 2023); Hagendorff (2024) studies language-model deception. These works study strategic interaction, but none tests the joint use of auctions, hidden-offer trade challenges, bluffing, discrete no-change payments, and portfolio-sensitive scoring.

CATTLE TRADE places these pressures in 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 supports both head-to-head ranking of policies and analysis of individual decisions (overbid frequency, self-bidding rate, bluff calibration).

3 CATTLE TRADE

3.1 THE CATTLE TRADE GAME

CATTLE TRADE adapts a 3–5 player tabletop 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. Appendix A details the turn structure and trade challenge mechanic; Table 1 compares CATTLE TRADE to existing game-based LLM benchmarks. CATTLE TRADE combines auctions, hidden-offer trade challenges (TCs), bluffing, and resource management within a single game over 50–60 turns.

Benchmark	Players	Info	Bargain	Bluff	Econ.	Hidden	Discrete$	Buy-right	Scoring	Turns
GTBench	2	Mixed	✓	✓	✗	✗	✗	✗	Binary	1–10
AvalonBench	5	Imperfect	✗	✓	✗	✗	✗	✗	Binary	5–10
Werewolf	7	Imperfect	✗	✓	✗	✗	✗	✗	Binary	5–15
MACHIAVELLI	1	Partial	✗	✗	✗	✗	✗	✗	Continuous	50+
Suspicion-Agent	2	Imperfect	✗	✓	✗	✗	✗	✗	Binary	10–20
ALYMPICS	5	Imperfect	✗	✗	✓	✗	✗	✗	Continuous	5–10
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. Hidden: face-down offers create information asymmetry within bargaining (an opponent must accept or counter without seeing the offered amount). Discrete$: payments use fixed-denomination money cards with no change given, so paying 60 with a 100 card costs 100. Buy-right: the auctioneer may pay the highest bid to keep the auctioned card rather than sell. Suite-style benchmarks (GameBench, GAMEBoT, lmgame-Bench) cover broad ranges of games whose properties vary by game; we cite them in §2 but omit from this table because per-property values would not be representative. CATTLE TRADE combines these properties through the TC mechanic and auction phase.

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 game 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 (sheep, goat and pig: 1,250 $\times$ 3 = 3,750) outscore two top quartets (cow and horse: 1,800 $\times$ 2 = 3,600), rewarding breadth over concentration and making the value of an additional quartet depend on the rest of the portfolio.

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 seven money cards

worth 90 coins (four 10s, one 50, two 0s). The 0 cards allow bluffing in TCs. An opponent who receives a face-down offer of three cards knows the card count but not the offered value.

On each turn, the active player either auctions the top card of the deck or initiates a TC (below). In an auction, the auctioneer either accepts the highest bid or exercises a *buy-right* (paying that amount to keep the card); with no bids, they get it free. High bids transfer more money to the auctioneer if accepted; low bids make the card cheaper for the auctioneer to keep. Overbidding one's holdings reveals wealth to all players and restarts the auction.

Instead of auctioning, the active player may initiate a TC 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. Both offers are then revealed and the money is exchanged: each player receives the other's coins, and the higher total wins. If each player holds at least two cards of the animal, the loser transfers two contested cards to the winner; otherwise the loser transfers one. Equal offers repeat the challenge; after three ties the initiator wins. Bluffing succeeds only if the opponent accepts without seeing the offer; any positive counter defeats a 0 bluff. The initiator must weigh the risk of losing animal cards against the cost of a larger offer.

Donkey cards distribute coins to all players on draw (50, 100, 200, 500 for the first through fourth donkey), injecting liquidity and creating phase transitions: early games are cash-constrained; late games permit larger bids.

3.2 Benchmark Design

The benchmark consists of a Python game engine, an LLM agent framework and an evaluation system. The engine implements the complete rules, validates every action, handles exact money-card accounting and logs all state transitions. It offers three auction modes; we use the *canonical* mode, which follows tabletop play with simultaneous bidding rounds. Agents receive natural-language observations (hand, visible cards, money, legal actions) and answer with a free-text analysis followed by a JSON move, each in its own tag. The move specifies exact bid amounts and the exact money cards of a TC offer (e.g., `offer_cards: [50, 10, 0]`), so the model constructs its bluffs itself. Each agent sees the full history of events it has observed and keeps private notes, which an additional call of the same model rewrites after new events. All agents receive the same system prompt (Appendix D: neutral rules plus "play optimally to maximize your expected score") without strategy hints. The engine applies the tabletop overbid penalty rather than correcting overbids, so they remain observable errors, and settles auction payments with the least overpay (Appendix C). We rate agents with TrueSkill (Herbrich et al., 2006) and with a margin score that also credits the size of a win (Section 4.1).

3.3 Code Agents

The engine is parameterised by set size, number of animal types, player count and starting hands, from a two-player duel with 3 animal types to a five-player game with 10; we use the standard four-player game with $k{=}4$ and 10 types. Three deterministic code agents serve as calibration baselines. TrackerAgent tracks revealed cards, opponent holdings and estimated wealth from all public events and bids just above an opponent's estimated budget. SetRaceAgent greedily pursues quartet completion and ignores animals with no path to a quartet. EconomyAgent caps its spending at a fraction of its wealth, avoids enriching cash-rich opponents and bluffs only against opponents unlikely to counter.

4 Experiments

4.1 Setup

Policies. The league contains 14 policies: 11 LLM policies built from eight models of five providers, and the three code agents of Section 3.3. Three Anthropic models enter at two reasoning-effort levels each: Claude Fable 5.1, Claude Opus 5.5 and Claude Sonnet 5, at *low* and *high* effort. The other five models enter at low effort only: GPT-6 Sol (OpenAI), Gemini 3.8 Flash and Gem-

ini 3.6 Flash (Google), GLM-5.3 (Zhipu AI) and DeepSeek 4.1 Flash (DeepSeek). Low effort is the default setting of the benchmark and the only level at which every model is evaluated. The high-effort policies form a paired ablation: the same model plays the same schedule at both levels, so a paired comparison estimates the effect of the configured reasoning effort on play. Code agents are TRACKERAGENT, SETRACEAGENT and ECONOMYAGENT.

Schedule. All 182 games are four-player games on the full deck (10 animal types, k=4) in canonical auction mode. A constraint solver fixed the schedule before the run. It uses 13 deck seeds with 14 games each. Within a seed, every policy plays four games, once in each seat, so deck order and seat position are balanced for every policy. Each policy plays 52 games, every pair of policies shares a table 12 times, and no table of four policies repeats. Games 1 to 91 form a balanced first half (every pair meets six times) that calibrates the rating scale before the second half is scored.

Uniform configuration. All LLM policies receive the same system prompt, memory and answer format (Section 3.2). Temperature is 1.0 for all models, a value some providers enforce for reasoning models, and the output limit is 32,768 tokens. If a move is missing, cannot be parsed or is illegal, up to two move-only retries at low effort are issued, and if these also fail the engine plays a default legal action. All models are called through LiteLLM on the providers' APIs, the two open-weight models on dedicated endpoints, and every call is logged with the returned model identifier, the effort sent and accepted, tokens and cost (Appendix Table 12).

Ratings. We report two performance measures. TrueSkill (Herbrich et al., 2006) treats each game as a four-player free-for-all ranked by final score and is updated in game order; we report μ and the conservative estimate $\mu - 3\sigma$. TrueSkill ignores margins, and Equation 1 produces margins that range over orders of magnitude. The second measure is a margin score. For players i and j at the same table with final scores S_i, S_j and ranks r_i, r_j,

$$u_{ij} = \operatorname{sgn}(r_j - r_i) + \lambda \tanh\left(\frac{\log(1+S_i) - \log(1+S_j)}{c}\right), \tag{2}$$

so finishing ahead of an opponent counts one point and the log-score margin adds at most λ. A policy's score in a game is the mean of u_{ij} over its three opponents, and its margin score is the mean over its games, with standard errors. The scale c is the median absolute log-score difference over all pairs in the valid games of the first half, frozen before any ranking is computed. We use λ=0.2 and report the ranking for $\lambda \in \{0, 0.1, 0.2, 0.3\}$, where λ=0 is a pure pairwise win count. The pairwise mean of u_{ij} between every two policies gives a 14×14 payoff matrix. A game is invalid if it ends with an error, reaches the turn limit or does not complete every quartet. Invalid games are excluded from every aggregate, and we list the policies each excluded game involved.

4.2 THE FRONTIER LEAGUE

Three policies lead the league and cannot be separated from each other: Fable 5.1 at high effort, Gemini 3.6 Flash and Opus 5.5 at high effort (Figure 2a,b). Fable 5.1 at high effort has the highest margin score (0.68, 95% CI $[0.51, 0.84]$) and wins 31 of its 52 games, but its lead over the next two is within the confidence intervals, and its rank interval spans places 1 to 3. At the other end, Sonnet 5 at low effort wins none of its 52 games. TrueSkill gives the same order up to two swaps between neighbours.

The ranking reflects the policies rather than the schedule. The two balanced halves of the league rank the policies almost identically (Spearman 0.92), seat position has no detectable effect, and the deck seed explains less than 1% of the variance of final scores. Removing any one policy moves no other policy by more than two places, and the pairwise results contain a single weak intransitive cycle. Appendix B.1 gives these checks, the pairwise payoff matrix and a first-place-only score that yields the same top six.

Most frontier policies clear the code-agent floor. Seven of the eleven LLM policies score clearly above TrackerAgent, the strongest code agent; three cannot be distinguished from it, and Sonnet 5 at low effort scores below it.

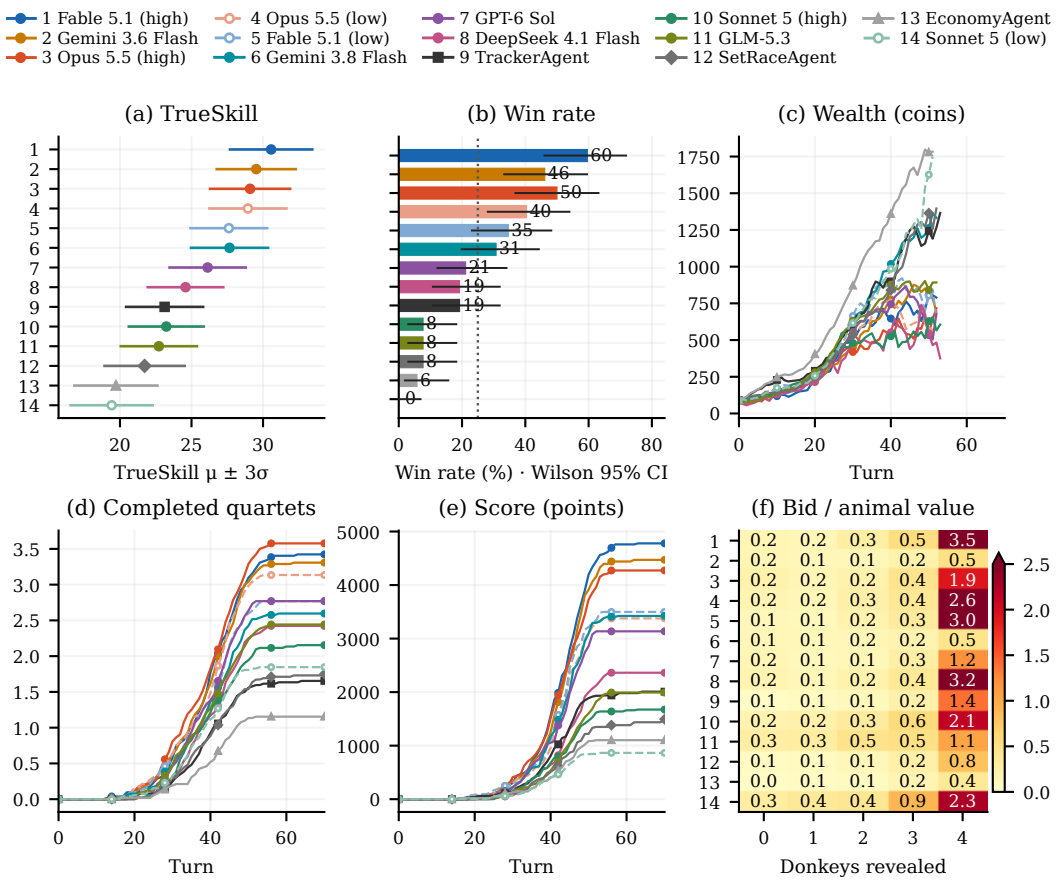


Figure 2: **The frontier league** (182 games, 52 per policy; numbers are the league ranks of Figure 1b). (a) TrueSkill $\mu \pm 3\sigma$ (σ is the posterior spread, not a confidence interval). (b) Win rate with Wilson 95% intervals; the dotted line is the 25% expected for an average player. (c) Mean wealth in coins over turns among running games, shown while at least 20 are running. (d, e) Mean completed quartets and score over turns. (f) Mean positive bid divided by the value of the auctioned animal, by number of donkeys revealed (colour capped at 2.5). Dark solid lines: high effort; light dashed lines: low effort; grey: code agents.

Cost does not order the field. Gemini 3.6 Flash costs \$0.61 per game, less than 4% of the \$16.82 of Fable 5.1 at high effort, and cannot be separated from it. Across the 11 LLM policies, cost per game and margin score are not significantly correlated (Spearman 0.40).

4.3 What Distinguishes Strong Play

Games are decided late. Until the first donkeys appear, all policies bid far below the value of the animals and almost no quartets are completed; quartets form in the second half of the game, and the final order is set there (Figure 2d,e). What matters is therefore how a policy turns the money that arrives with the donkeys into completed sets. We profile each policy with the behavioural metrics of Appendix B.3 and with financial profiles (Appendix B.1), and read them together rather than one at a time.

Two ways to win. The leading policies reach the same number of quartets by different routes. Fable 5.1 and Opus 5.5 act as traders: they start many trade challenges, win most of them, gain animals from their opponents on balance, and bid well above an animal's value late in the game. The two Gemini models act as patient buyers: they bid close to half of an animal's value even at the end, almost never bluff, and convert money into score more efficiently than any other policy. Neither aggressive bidding nor bluffing is therefore required for strong play; what the leaders share is that

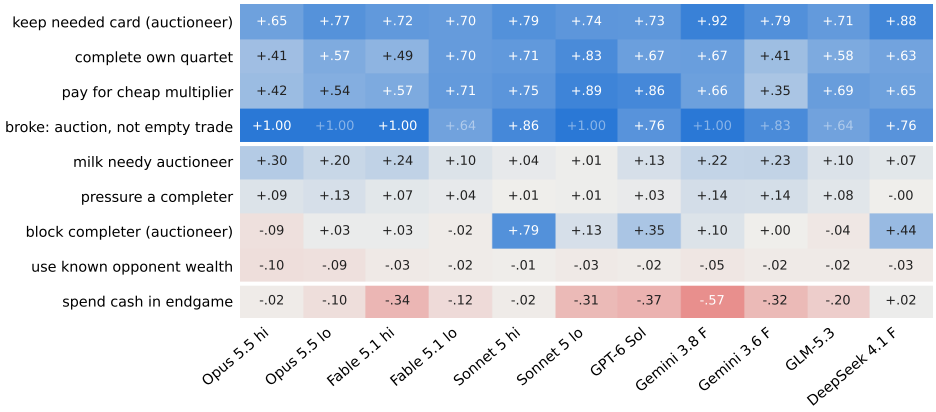


Figure 3: **Behaviour profiles in the league.** Each decision in which an axis applies scores $+1$ if the model takes the opportunity, -1 if it shows the weakness and 0 otherwise; cells show the mean per model. Top block: the model's own quartets and money. Middle block: exploiting opponents. Bottom row: using money in the endgame. Faded cells rest on fewer than 15 decisions. Axes are defined in Appendix Table 13.

they complete many sets and spread their money over many purchases instead of concentrating it, and the leaders of the earlier tier share the same spending pattern. Across the 11 LLM policies, completed quartets, trade challenges started and capital efficiency remain associated with rank after correction for multiple tests, while bid aggressiveness shows no detectable association.

Two ways to lose. Weak policies fail in two opposite ways. The code agents and Sonnet 5 at low effort hoard: they leave much of the donkey money unspent and end games holding a third to half of the money at the table, which scores nothing, and they start fewer trade challenges than the leaders. DeepSeek 4.1 Flash and GLM-5.3 spend instead: they make large single payments relative to their wealth and lose animals in trade challenges, so their spending does not turn into sets. The combination that separates the field is thus not how much a policy spends but whether its spending is coherent with a plan for its sets.

Competent on their own portfolio, blind to opponents. To see which skills are in place, we score every decision against situations with a clear right move, such as keeping a needed card as auctioneer or raising the price of a card that the auctioneer needs (Figure 3; Appendix E). Every model plays its own portfolio well: it keeps cards it needs, completes its quartets and buys cheap multipliers. Almost none exploits its opponents: models rarely bid up a card the auctioneer needs, rarely pressure a player one card from a quartet, and do not use an opponent's money when it is known. Blocking a completer as auctioneer is the only such skill that one model, Sonnet 5 at high effort, uses consistently. The one weakness nearly all models share concerns money in the endgame: when a quartet is decided in the last draws, they let it go while still able to pay, although money is worth nothing once the game ends. The differences between frontier models therefore lie less in basic competence than in how consistently they combine their skills over the game, and the interactive part of strategy, reasoning about what opponents need and can pay, is largely missing. Individual traces add errors of valuation: in one game Sonnet 5 at low effort passes on the last cow because it values its single donkey quartet at "500x4=2000 pts", although it scores 500 (Equation 1).

4.4 Reasoning Effort

High effort clearly helps one of the three models. Comparing the two effort levels of each model seed by seed, and leaving out the games in which both levels share a table, Sonnet 5 gains $+0.56$ in margin score (95% CI $[0.41, 0.71]$) and does better at high effort against every other policy. The gains of Fable 5.1 and Opus 5.5 cannot be distinguished from zero.

The effort setting does not mean the same thing across models. High effort multiplies the reasoning tokens per call by 96 for Sonnet 5 but only by 3 for Opus 5.5, and at low effort the providers range

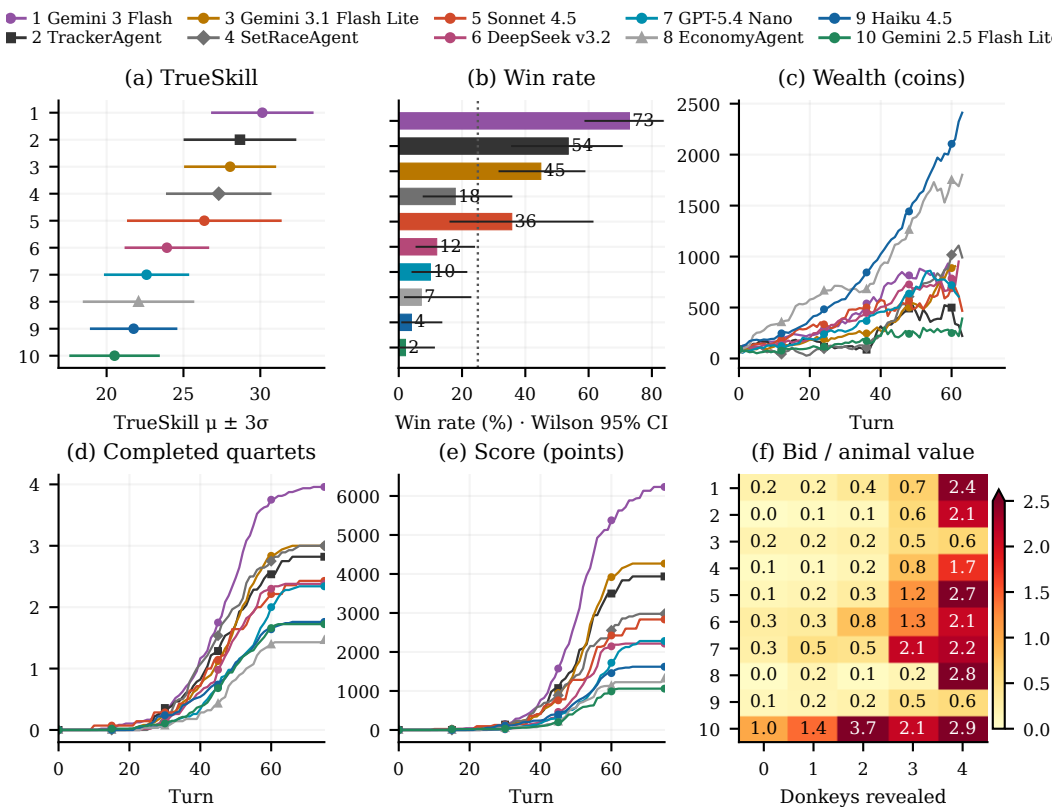


Figure 4: **The earlier cost-efficient tier**, with panels (a) to (f) as in Figure 2 (98 canonical-auction games: 70 all-LLM games and 28 games of one LLM against the three code agents). Row numbers order the 10 policies by TrueSkill μ within this tier. Unlike the league schedule, this slice is not balanced, and policies play different numbers of games.

from almost no reasoning (Sonnet 5) to about 2,000 reasoning tokens per call (DeepSeek 4.1 Flash). The official low-effort ranking therefore compares each model at its provider's implementation of that setting (Appendix Table 5).

4.5 Two Tiers of Models

The earlier tier of seven cost-efficient LLMs (242 games, Appendix B.2) shares only the code agents with the league, and models and configuration changed together (Appendix Table 6), so we compare the tiers descriptively rather than on one scale. Figure 4 shows the earlier tier in the form of Figure 2. TrackerAgent ranked above five of the seven cost-efficient LLMs and ranks below eight of the eleven frontier policies. In both tiers, completed quartets and capital efficiency follow the rank, while trade-challenge activity does so only in the frontier league. Overbidding and wasteful self-bidding, the failure modes documented for the cost-efficient models, occur at similar rates in both tiers.

4.6 Cost and Format Compliance

The league made 73,643 decisions in 7 hours at a cost of $2,186. Format problems were rare: 0.08% of first answers failed to parse or were illegal, all from GLM-5.3 and DeepSeek 4.1 Flash, and the engine substituted 18 default moves in total.

5 DISCUSSION

Frontier models have overtaken the simple heuristics that beat most cost-efficient models, but not uniformly. The spread within the frontier field is larger than the gap between its best members, and neither price nor reasoning budget predicts where a policy lands: the cheapest of the leading models cannot be separated from the most expensive one, and the weakest policy in the league is a current frontier model at its provider's low effort setting. For evaluation this means that a model name is not a sufficient description of an agent; the effort setting and the provider's implementation of it are part of the policy.

The largest gap is interactive: models play their own portfolio competently but rarely act on what their opponents need or can pay, which is the part of strategic reasoning that isolated benchmarks cannot probe. The behavioural results show what distinguishes strong play in this game. Strong policies turn money into completed quartets efficiently and use trade challenges actively once the auction phase thins out, while how aggressively they bid is not related to their rank. These are associations across 11 policies and do not establish causes; a policy that is ahead may, for example, start more trade challenges because it can afford to. The failure modes documented for cost-efficient models, overbidding and wasteful self-bidding, have not disappeared at the frontier, and the traces add errors of valuation and money left unspent at the end of a game. The code agents are useful as a shared reference across tiers but not as a fixed scale, since their results depend on the field and on the bid grid they play under.

Limitations. Each policy is rated from 52 games of one schedule, the top three policies are not separated, and strength is measured against this field only. All models play with one prompt, memory configuration and answer format at temperature 1.0, and only three models are evaluated at two effort levels. No policy is told to check its wealth before bidding, while the code agents act on structured data with exact arithmetic, so valuation errors may reflect numerical parsing or working memory as well as strategy. Whether the failure modes transfer to other economic settings is untested.

6 CONCLUSION

We introduced CATTLE TRADE, a multi-agent benchmark integrating auctions, hidden-offer TCs, bluffing, and resource management. Across 424 games in two tiers, a balanced 182-game league of eight frontier LLMs and an earlier 242-game evaluation of seven cost-efficient LLMs, both played against the same three code agents, success is associated with spending efficiency in both tiers and, in the frontier league, with the use of trade challenges, and shows no detectable association with bidding aggressiveness. Most LLM policies now outperform the strongest deterministic heuristics, which outperformed most cost-efficient models, yet one frontier configuration still scores below the strongest of them, and a model costing under 4% as much per game cannot be distinguished from the strongest policy. The structured logs show *where* models fail (wasteful self-bidding spirals, valuation errors, reluctance to initiate trade challenges), failure modes invisible to aggregate evaluations and consistent with concrete gaps in state tracking, resource-constraint reasoning, and opponent modelling that static benchmarks do not measure. More broadly, benchmarks of this kind may become increasingly important for evaluating agents beyond static reasoning or single-turn task success. They make it possible to test whether models can sustain strategic coherence over time, manage resource constraints, and adapt interactively in multi-agent environments with conflicting incentives. This matters for prospective deployments in economic and organizational settings, where failures often arise not from the absence of any single capability, but from the inability to coordinate multiple capabilities across extended interactions. In this sense, CATTLE TRADE is a step toward evaluating agentic competence under more realistic conditions of strategic interaction.

Several directions follow. Our effort ablation covers three models at two levels; evaluating all models at their full reasoning budgets would test whether the remaining failures reflect fundamental limitations or the providers' low-effort settings. Enabling inter-agent chat would introduce the meta-communication layer present in human play (coalition signalling, persuasion, relationship management) that the current structured-action format excludes. Self-play fine-tuning of open-weight models could test whether strategic failure modes such as self-bidding spirals can be trained away. Finally, benchmarking against human players would calibrate whether LLM failures reflect genuinely hard strategic problems or errors that novice humans also avoid.

AI USE STATEMENT

Large language models are the object of study in this work: the eleven LLM policies of Section 4 were queried through their providers' APIs as game-playing agents. In addition, we used generative AI assistants (Claude models in Claude Code, and OpenAI models in ChatGPT and Codex) for the following research tasks. *Literature research*: finding related work and checking every reference against its source for correct bibliographic data and for support of the claim that cites it. *Implementation*: writing and debugging parts of the game engine, agent framework, league infrastructure and analysis code, including the constraint program that generates the schedule. *Running experiments*: launching and monitoring the league runs and diagnosing failures. *Analysis*: computing ratings and behavioural metrics, producing figures, and drafting interpretations of the results, which the authors checked against the game logs. *Writing*: drafting and editing parts of the text and formatting references. The research question, the benchmark design, the policy pool, the effort levels, the schedule constraints and the rating method were chosen by the authors. We have reviewed all AI-assisted work, verified the reported numbers against the logs, and take full responsibility for the content of the paper.

ETHICS STATEMENT

The benchmark studies deception only inside a game whose rules make bluffing a legal move, and no human participants were involved. Models that learn to bluff or to exploit opponents in negotiation could transfer these behaviours to economic settings with real counterparties; we report such behaviours as measurements and release no method for improving them. CATTLE TRADE is a research implementation of a commercial card game and is not affiliated with its publisher.

REPRODUCIBILITY STATEMENT

Section 3.2 and Appendices A, C and D describe the game engine, code agents, prompts and full league configuration (Table 12). The league schedule is fixed in advance and balanced by construction (Section 4.1), and the rating scale is frozen from the first half before any ranking is computed. Appendix B defines every metric and reports the per-policy results from which the figures are drawn.

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. 3

Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890, 2019. 3

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 Allen, Roman Hauksson, Grace Sodunke, Suhas Hariharan, Carlson Cheng, Wenjie Li, Joshua Clymer, and Arjun Yadav. 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 Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. GTBench: Uncovering the strategic reasoning capabilities of LLMs via game-theoretic evaluations. In *Advances in Neural Information Processing Systems*, volume 37, pp. 28219–28253, 2024. 1, 2

Kanishk Gandhi, Dorsa Sadigh, and Noah D. Goodman. Strategic reasoning with language models. *arXiv preprint arXiv:2305.19165*, 2023. 1, 3

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. In *Conference on Language Modeling (COLM)*, 2024. 2

Thilo Hagendorff. Deception abilities emerged in large language models. *Proceedings of the National Academy of Sciences*, 121(24):e2317967121, 2024. 3

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, pp. 569–576. MIT Press, 2006. 4, 5, 23

Sirui Hong, Mingchen Zhuge, Jiaqi 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, pp. 64992–65024, 2025. 3

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? End-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2443–2453, 2017. 3

Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. AvalonBench: 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. 3

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. 3

Alexander Pan, Jun Shern Chan, Andy Zou, Nathaniel Li, Steven Basart, Thomas Woodside, Hanlin Zhang, Scott Emmons, and Dan Hendrycks. Do the rewards justify the means? Measuring trade-offs between rewards and ethical behavior in the MACHIAVELLI benchmark. In *International Conference on Machine Learning*, volume 202 of *PMLR*, pp. 26837–26867, 2023. 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

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 CATTLE TRADE.

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. Canonical mode runs as iterative *call rounds*: in each round, all non-auctioneer (non-eliminated) players submit a bid intent (a number or *pass*) simultaneously. The highest valid bid in the round becomes the current standing bid (ties broken by a per-auction random priority order; see Appendix C). The auction continues into further rounds (bidders may raise across rounds) and ends the first time all eligible bidders pass while a standing bid exists. If *no* bid is placed in round 0, the engine runs one confirmation round; if that round is also empty, the auctioneer receives the card for free. This two-round rule prevents a empty initial call from handing the card to the auctioneer. Once the auction ends, the auctioneer sees the winning 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 a player bids more than they have, their wealth is revealed to all players and the auction restarts; a second overbid by the same player in that auction eliminates them from it.

Trade challenge 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 cards of the contested animal to the initiator; 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. If each side holds at least two cards of the animal, the loser transfers two contested cards to the winner; otherwise the loser transfers one. 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 TC 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 LEAGUE RESULTS

Table 3 lists the ratings, outcomes and costs of all 14 policies, and Figure 5 the pairwise payoff matrix. Token counts are means per decision call as reported by the providers; reasoning tokens are the provider-reported hidden reasoning and exclude the visible analysis. Code agents make no API calls.

Stability of the ranking. Ranked separately on the two balanced halves of 91 games, the orders agree with a Spearman correlation of 0.92 (0.86 over the 11 LLM policies). Removing any one

Policy	Margin ± SE	TrueSkill $\mu \pm \sigma$	Win %	Mean rank	Mean score	Median score	Rank H1 / H2	Cost / game ($)	Output tok. / call	Reasoning tok. / call
Fable 5.1 (high)	$+0.68 \pm 0.09$	30.6 ± 1.0	59.6	1.62	4,778	4,450	2 / 1	16.82	1,575	298
Gemini 3.6 Flash	$+0.55 \pm 0.09$	29.5 ± 0.9	46.2	1.79	4,470	3,690	3 / 3	0.61	516	349
Opus 5.5 (high)	$+0.55 \pm 0.10$	29.1 ± 0.9	50.0	1.81	4,272	3,930	1 / 4	4.71	855	130
Opus 5.5 (low)	$+0.42 \pm 0.10$	28.9 ± 0.9	40.4	1.94	3,376	2,860	6 / 2	3.30	440	40
Fable 5.1 (low)	$+0.32 \pm 0.10$	27.6 ± 0.9	34.6	2.08	3,498	2,865	4 / 6	8.80	436	23
Gemini 3.8 Flash	$+0.30 \pm 0.10$	27.7 ± 0.9	30.8	2.10	3,423	3,300	5 / 5	0.46	255	27
GPT-6 Sol	$+0.09 \pm 0.11$	26.1 ± 0.9	21.2	2.40	3,136	2,650	7 / 7	1.00	180	75
DeepSeek 4.1 Flash	-0.08 ± 0.11	24.6 ± 0.9	19.2	2.60	2,362	1,790	8 / 8	0.52	2,323	2,168
TrackerAgent	-0.27 ± 0.13	23.1 ± 0.9	19.2	2.85	2,008	1,620	10 / 10	–	–	–
Sonnet 5 (high)	-0.27 ± 0.10	23.2 ± 0.9	7.7	2.83	1,674	1,430	11 / 9	3.21	1,287	213
GLM-5.3	-0.32 ± 0.10	22.7 ± 0.9	7.7	2.92	1,993	1,490	9 / 11	0.77	280	111
SetRaceAgent	-0.52 ± 0.10	21.7 ± 0.9	7.7	3.17	1,494	1,000	12 / 12	–	–	–
EconomyAgent	-0.70 ± 0.10	19.7 ± 1.0	5.8	3.38	1,102	800	13 / 13	–	–	–
Sonnet 5 (low)	-0.75 ± 0.07	19.4 ± 1.0	0.0	3.44	865	650	14 / 14	1.84	222	2

Table 3: League results per policy (182 games, 52 per policy), sorted by margin score (λ=0.2) with its standard error over games. TrueSkill: posterior mean and standard deviation (not a confidence interval). Win %: share of games finished first (25% is the expected share at a four-player table). Mean rank: finishing position averaged over games; scores are final game points. Rank H1 / H2: margin-score rank within the first and second balanced half (91 games each). Cost is logged US dollars per game; token counts are means per call.

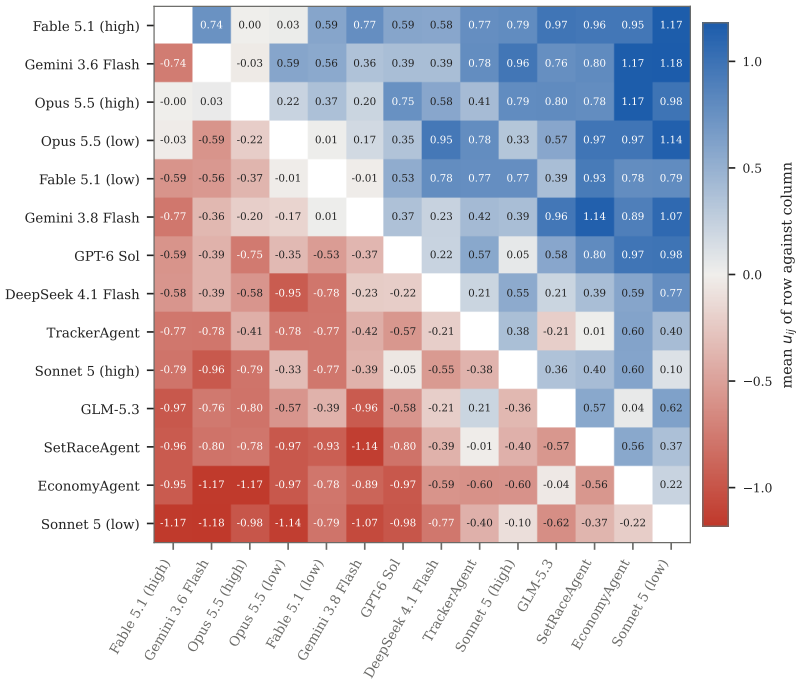


Figure 5: Pairwise payoff matrix. Each cell is the mean of u_{ij} (Equation 2, λ=0.2) of the row policy against the column policy over the 12 games in which the two share a table; rows and columns are sorted by margin score. Blue: the row policy finishes ahead on average.

Policy	η median	τ median	Quartets / game	TC init. / game	TC win init. %	Pure-0 bluff %	Overbid %	Buy-right %	Jump $\geq$50 %	Spirals total	WASTE %
Fable 5.1 (high)	1.32	0.48	3.42	2.75	79	11.0	0.4	27	46	13	8
Gemini 3.6 Flash	1.65	0.49	3.31	3.25	73	2.2	0.0	27	31	25	20
Opus 5.5 (high)	1.37	0.42	3.58	2.94	84	3.4	0.5	32	36	10	10
Opus 5.5 (low)	1.23	0.47	3.13	2.65	72	1.6	0.1	29	36	4	25
Fable 5.1 (low)	1.03	0.57	2.77	2.00	80	8.9	0.1	25	42	8	38
Gemini 3.8 Flash	1.56	0.51	2.60	2.52	67	2.2	0.0	35	33	29	7
GPT-6 Sol	1.07	0.36	2.77	2.23	68	6.2	0.5	30	31	11	45
DeepSeek 4.1 Flash	0.58	0.40	2.42	1.88	58	15.9	2.0	26	38	14	14
TrackerAgent	0.82	0.32	1.65	2.12	61	16.0	0.0	23	5	147	0
Sonnet 5 (high)	0.55	0.52	2.15	1.98	61	11.0	0.7	33	29	82	7
GLM-5.3	0.65	0.48	2.44	2.17	58	4.3	0.5	18	32	101	15
SetRaceAgent	0.64	0.24	1.75	1.98	53	15.7	0.0	28	0	254	17
EconomyAgent	0.75	0.35	1.15	1.52	35	38.1	0.0	21	0	217	6
Sonnet 5 (low)	0.56	0.41	1.85	0.85	70	11.4	0.2	24	11	121	7

Table 4: Behavioural metrics per policy in the league, sorted by margin score. η and τ as in Appendix B.3 (medians over games). TC win init.: share of initiated TCs won by the initiator. Pure-0 bluff: share of TC offers made only of 0-value cards. Overbid: share of positive bid intents that exceed the bidder's money. Buy-right: share of the policy's auctions as auctioneer settled by buy-right. Jump: share of positive bids raising the standing price by at least 50 coins. Spirals: total count over the policy's 52 games; WASTE: share of those spirals classified as WASTE (Appendix B.3).

policy and recomputing the margin scores moves every other policy by at most two places. Win rates by seat range from 21% to 29% ($\chi^2(3)$=2.40, p=0.49; Figure 13). In an analysis of variance of final scores, the policy explains 27% of the variance, the deck seed 0.4% and their interaction 17% (p=0.27). The pairwise payoff matrix contains one intransitive triple among 364 (GLM-5.3 over TrackerAgent over Sonnet 5 at high effort over GLM-5.3), with a weakest edge of 0.21. The ranking is identical for $\lambda \in \{0.1, 0.2, 0.3\}$; at λ=0 TrackerAgent and Sonnet 5 at high effort tie. Fable 5.1 at high effort ranks first in 73% of game-bootstrap resamples, and its margin score exceeds those of Gemini 3.6 Flash and Opus 5.5 at high effort by 0.13 (95% CI $[-0.13, 0.38]$) and 0.14 ($[-0.14, 0.41]$). Seven LLM policies score above TrackerAgent with bootstrap intervals of the difference that exclude zero; DeepSeek 4.1 Flash, Sonnet 5 at high effort and GLM-5.3 cannot be distinguished from it, and Sonnet 5 at low effort is below it. Across the 11 LLM policies, the Spearman correlation between cost per game and margin score is 0.40 (95% interval $[-0.33, 0.85]$).

Rank uncertainty and a first-place score. Figure 6 shows 95% intervals of each policy's rank from 2,000 bootstrap resamples of games and of deck seeds. As a check that the margin score does not reward second and third places too much, we also score each game by first place alone: 1 for the winner (split on ties) plus $\lambda \tanh(\Delta/c)$, where Δ is the log-score difference to the best opponent. For $\lambda \in \{0, 0.2, 0.5\}$ this score yields the same top six and the same last place as the margin score; Opus 5.5 at high effort and Gemini 3.6 Flash exchange places 2 and 3, and TrackerAgent, DeepSeek 4.1 Flash, GLM-5.3, Sonnet 5 at high effort and SetRaceAgent reorder among themselves, all within overlapping intervals.

Reasoning effort. Table 5 gives the effort comparison of Section 4.4. The primary estimate excludes the 12 games in which both levels of a model share a table and pairs the 13 deck seeds; the common-opponent estimate compares the two levels against each of the 12 other policies. Including the shared tables raises the Fable 5.1 estimate to $+0.36$ (p=0.008), because its direct encounters then count for both levels; we therefore report the exclusion.

Economics, auctions and trade challenges. Figure 7 plots capital efficiency against TC tightness, and Figure 8 the price of auctions over the game. Figures 9 and 10 show trade-challenge activity over the game and bluffing and calling per policy. Of 1,604 trade challenges, the target accepted 39% of first offers; 10.5% of offers consisted only of 0-value cards. Figure 11 summarises six strategy metrics per policy, Figure 12 the cost per completed quartet by animal, and Figure 13 the pooled seat test.

Financial profiles. Treating each player as a trader, Figure 14 standardises eight financial measures within each tier: share of the table's money held at the end, share of donkey payouts spent,

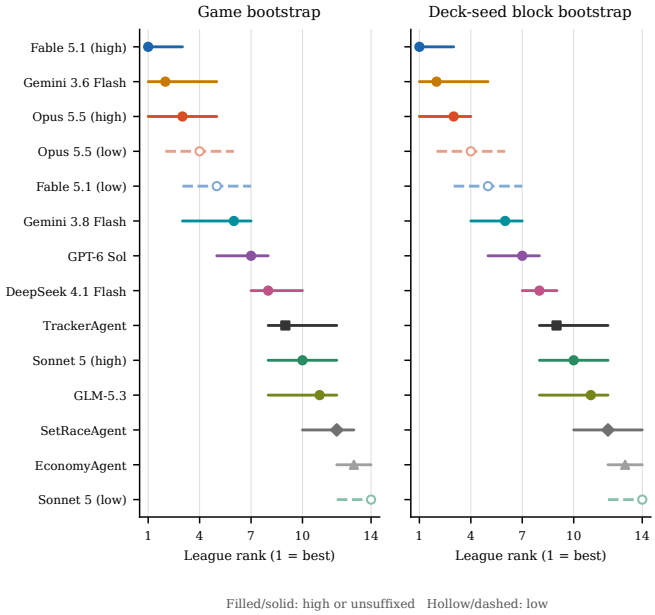


Figure 6: 95% intervals of league rank from bootstrap resamples of games and of deck seeds (2,000 each).

Model	Seed Δ [95% CI]	Holm p	Common opp. Δ	Positive	Direct W-L-T	Reasoning tok. ratio	Cost / game ($)
Fable 5.1	$+0.18$ $[-0.13, +0.50]$	0.68	$+0.22$	7/12	9-3-0	13.0	8.80 → 16.82
Opus 5.5	$+0.07$ $[-0.23, +0.37]$	0.84	$+0.06$	7/12	7-5-0	3.3	3.30 → 4.71
Sonnet 5	$+0.56$ $[+0.41, +0.71]$	< 0.001	$+0.56$	12/12	6-5-1	95.6	1.84 → 3.21

Table 5: High minus low reasoning effort. Seed Δ: mean over 13 deck seeds of the difference in mean margin score, excluding games in which both effort levels of the model play, with a t-interval; Holm correction over the three Wilcoxon tests. Common opp.: mean difference of pairwise margin against the 12 other policies, and the number of those opponents against which high effort does better. Direct W-L-T: wins, losses and ties of the high- against the low-effort policy in the games where both play. Reasoning tok. ratio and cost compare high with low effort.

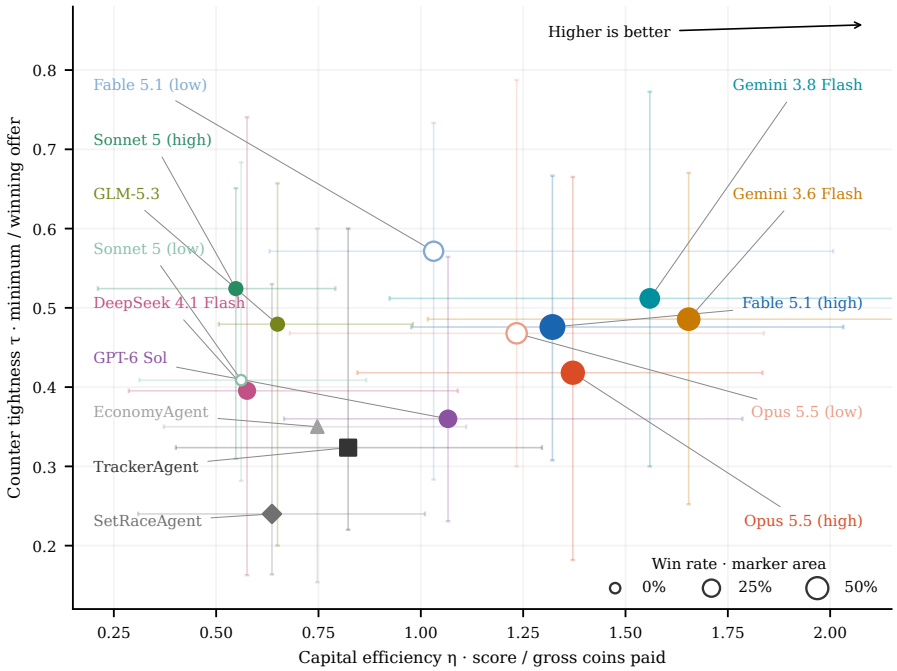


Figure 7: Capital efficiency η against TC tightness τ (medians over games, IQR bars); marker area is proportional to win rate. Both metrics describe play; they do not by themselves establish why a policy wins.

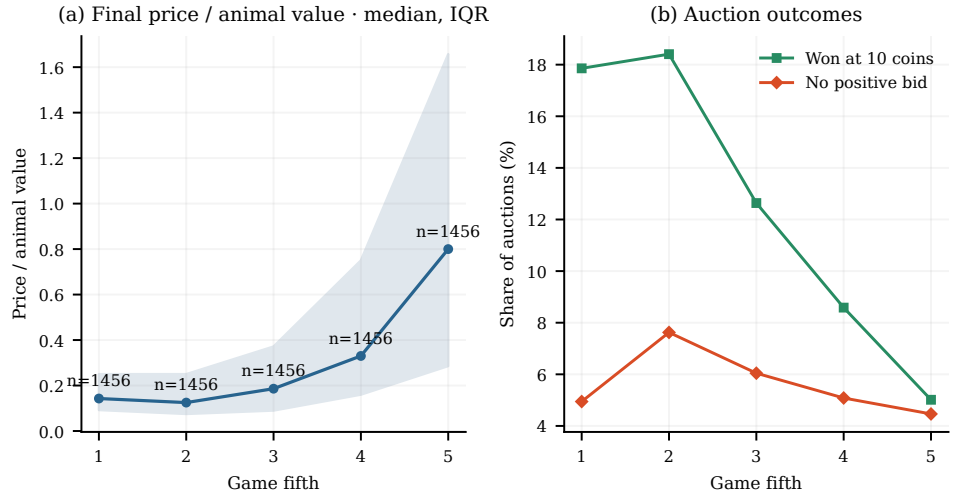


Figure 8: Auctions over the game, pooled over policies: final price divided by animal value by fifth of the game (median and IQR), and the shares of auctions won at 10 coins and without any bid.

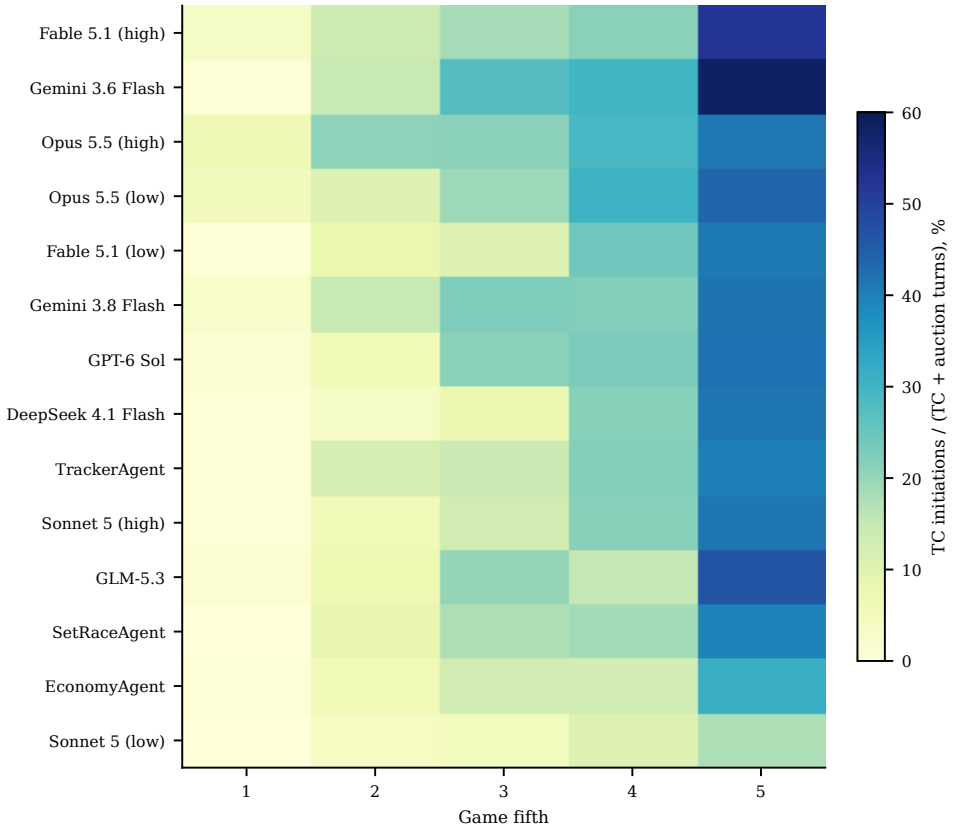


Figure 9: Trade-challenge initiation by fifth of the game: initiations as a share of the policy's own turns (auction or trade challenge).

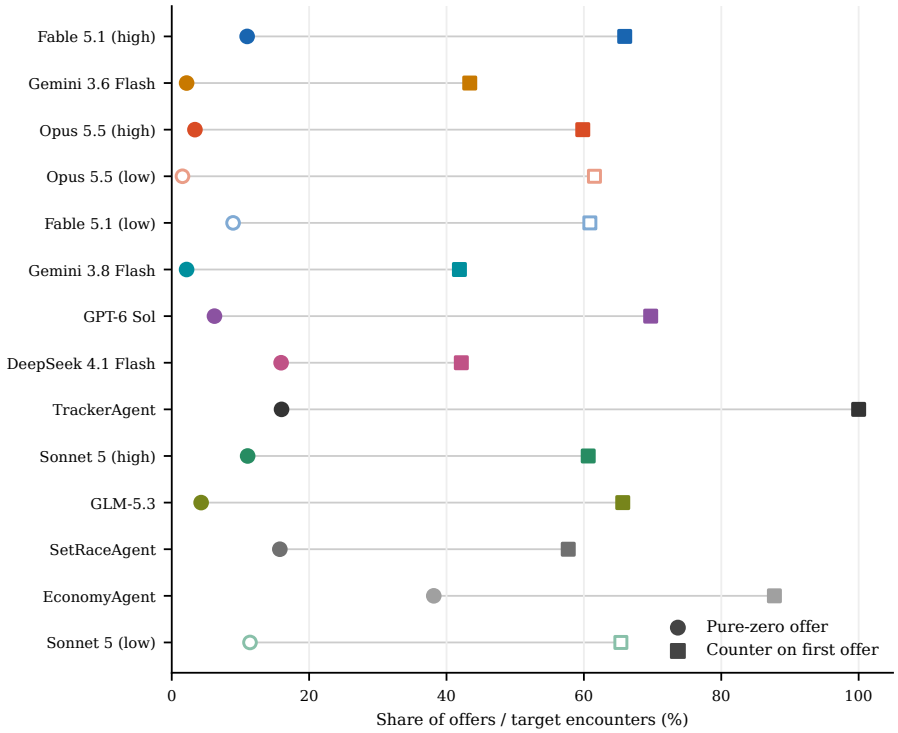


Figure 10: Bluffing and calling per policy: share of the policy's offers made only of 0-value cards (circles), and share of first offers it counters as a target (squares).

largest payment relative to wealth, concentration of spending, auction price relative to quartet value, surplus as auctioneer, net animals gained in trade challenges, and the share of games without a quartet. The profiles are descriptive and are not compared across tiers.

Tiers. Table 6 lists the configuration differences between the tiers, and Figure 15 each LLM's mean pairwise margin against the code agents in the games they share, within its own tier.

Setting	Cost-efficient tier (242 games)	Frontier league (182 games)
Models	Haiku 4.5, Sonnet 4.5, Gemini 2.5 Flash Lite, Gemini 3 Flash, Gemini 3.1 Flash Lite, GPT-5.4 Nano, DeepSeek v3.2	Fable 5.1 and Opus 5.5 and Sonnet 5 (high and low), GPT-6 Sol, Gemini 3.8 and 3.6 Flash, GLM-5.3, DeepSeek 4.1 Flash
Temperature	0.1	1.0
Output limit	4,096 tokens (16,384 in four games)	32,768 tokens
Answer format	JSON only	analysis, then JSON move
Memory	recent events and notes	full event history and notes
Reasoning effort	low	low; high in addition for three models
Tables	all-LLM tables and one LLM against three code agents	balanced schedule of 14 distinct policies
Code-agent bids	any amount	rounded down to multiples of 10

Table 6: Configuration of the two tiers. Both use the full deck, canonical auctions and the same neutral system prompt.

B.2 Earlier Evaluation of Cost-Efficient Models

The figures and tables below come from an earlier 242-game evaluation of seven cost-efficient LLMs (Claude Haiku 4.5, Claude Sonnet 4.5, Gemini 2.5 Flash Lite, Gemini 3 Flash, Gemini 3.1 Flash Lite, GPT-5.4 Nano and DeepSeek v3.2) against the same three code agents, played at temperature

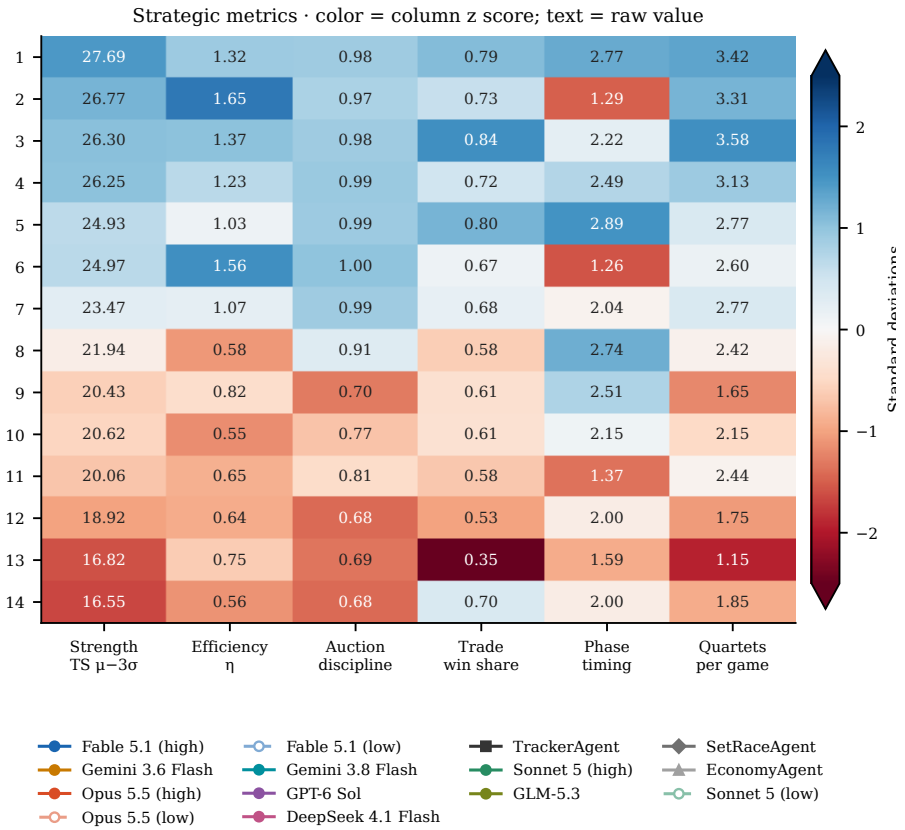


Figure 11: Six strategy metrics per policy, z-scored across policies (colour) with raw values printed; rows sorted by margin score.

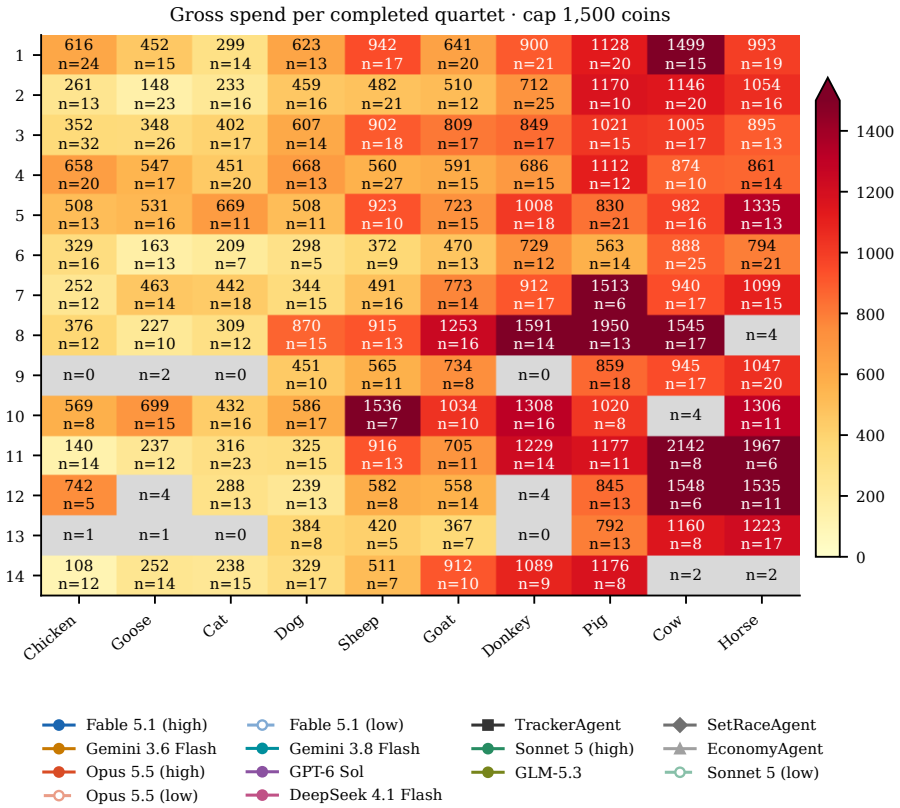


Figure 12: Gross coins paid per completed quartet, by animal and policy; cells based on fewer than five quartets are greyed and show the count.

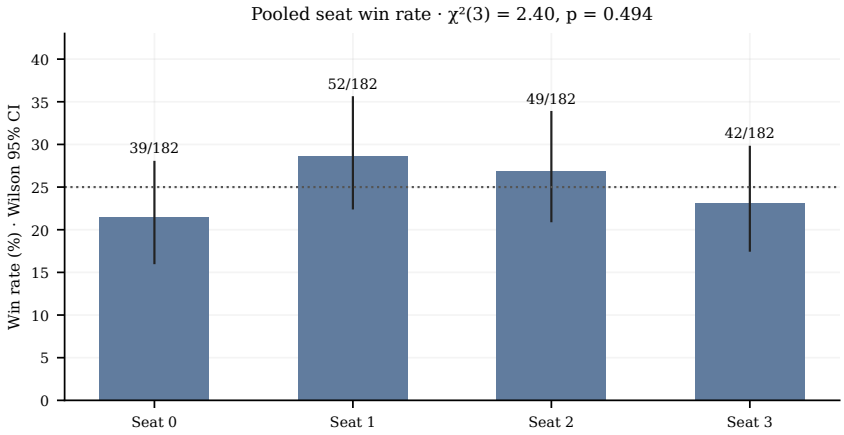


Figure 13: Win rate by seat, pooled over policies (182 games per seat) with 95% intervals and a χ^2 test against 25%.

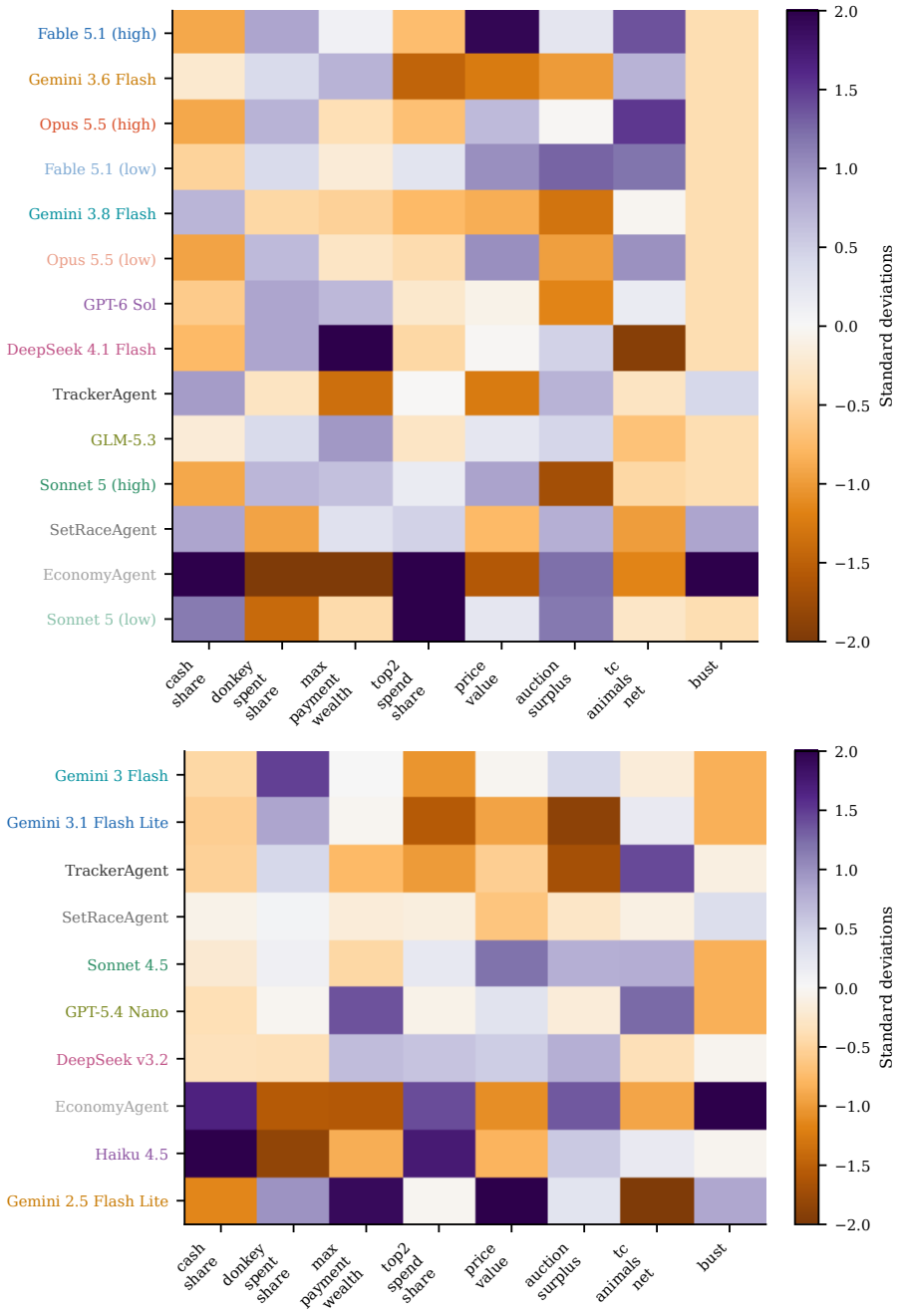


Figure 14: Financial fingerprints of the frontier league (top) and the cost-efficient tier (bottom). Columns are the eight measures listed above, in that order. Each column is standardised across the policies of its own tier; zero is the tier mean and colour shows standard deviations from it.

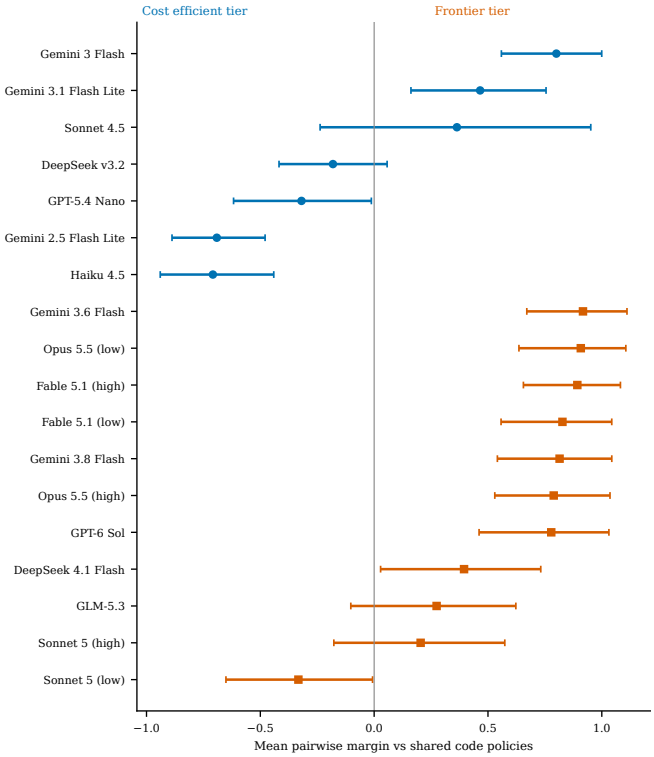


Figure 15: Mean pairwise margin of each LLM against the three code agents in the games where they met, within its own tier, with bootstrap 95% intervals. The two groups differ in models, configuration and schedule, so policies should be compared only within a group; the values do not place the two tiers on one scale.

0.1, low reasoning effort, a 4,096-token output limit and JSON-only answers. Table 7 uses all 242 games; the figures and the other tables use its 98-game canonical slice (70 pure-LLM games and 28 games of one LLM against the three code agents). Table 7 uses the definitions below. The figures of this subsection were produced with the earlier analysis, whose capital efficiency counts only the final round of a trade challenge; the values can therefore differ slightly from the table for games with tied rounds.

B.3 Metric Definitions

The per-agent metrics in Section 4.3 and the figures and tables below are computed from raw game logs. TC abbreviates trade challenge. $\mathcal{Q}$ denotes the set of completed quartets and v_q the quartet value (Equation 1).

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

Capital efficiency η. $\eta = S/\mathrm{out}$ per game, where out is gross outflow: every auction payment including buy-right payments, and in trade challenges both sides' offers in every counter round, including tied rounds; an accepted challenge pays its offer. Reported as the median of the per-game ratios; games without outflow are excluded.

TC bargaining 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 $\tau = \sum_k(\ell_k + 10)/\sum_k w_k$, size-weighted per game (summing over counter-wins k). Money cards step by 10, so $\ell_k + 10$ is the minimum bid the winner needed to win; τ is bounded by 1 because $w_k \geq \ell_k + 10$. τ=1 = winner paid only the minimum increment; $\tau\to 0$ = winner crushed a small counter with a much larger offer.

Bid aggressiveness. Mean of bid$/v_q$ over the agent's individual positive bid intents, where v_q is the quartet value of the auctioned animal. Late aggressiveness uses the intents placed after the fourth donkey has been revealed.

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

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

Bluff % (pure-zero offer rate). Fraction of the agent's TC offers, as challenger or as countering target, that consist only of 0-value money cards.

Self-bid %. Fraction of the agent's auction bids placed in rounds with no competing bid since this agent's last bid (i.e., the agent is raising while no other bidder has acted in the interim). Equivalently in canonical mode, the agent was the current high bidder when it bid again.

Overbid %. Logged overbids (bids exceeding the bidder's total money) divided by the agent's positive bid intents, pooled over its games. An overbid triggers the canonical-mode wealth revelation (Section 3.2).

Spirals and WASTE %. A spiral is a run of at least three consecutive bidding rounds in which one bidder is the only positive bidder and raises its own bid; Table 8 defines the five outcome classes. WASTE % is the share of a policy's spirals classified as WASTE. A spiral is an observed pattern, not proof of a mistake.

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. The paper ranks by posterior mean μ; $\mu_c = \mu - 3\sigma$ is reported as a conservative skill estimate where useful.

In the 98-game canonical slice, score variance is high: Gemini 3 Flash ranges from 0 to over 10,000 (mean 6,232, std 4,026), reflecting the multiplicative scoring formula's sensitivity to quartet count. Gemini 2.5 Flash Lite shows the highest relative variance (std 1,283 on mean 1,060). Scores vary substantially within each agent across games; these statistics do not separate deck-order and seat-position effects.

Figure 16 provides detailed auction and TC metrics across the ten agents. The cross-play win-rate heatmap (bottom panel) shows Gemini 3 Flash winning the majority of games against every

Policy	Margin	Games	η median	τ median	Quartets / game	TC init. / game	TC win init. %	Pure-0 bluff %	Overbid %	Buy-right %	Jump $\geq$50 %	Spirals / game	WASTE %
Gemini 3 Flash	+0.78	72	1.78	0.37	3.94	4.67	72	10.6	0.1	23	38	1.22	15
Gemini 3.1 Flash Lite	+0.50	73	1.51	0.35	3.08	5.56	61	8.2	0.4	36	38	0.45	21
TrackerAgent	+0.24	172	1.07	0.26	2.60	4.69	56	20.8	0.0	32	6	1.74	0
SetRaceAgent	+0.12	172	1.20	0.24	2.68	3.55	63	16.9	0.0	28	0	4.55	22
Claude Sonnet 4.5	+0.10	14	0.75	0.30	2.43	3.29	72	15.7	0.4	22	49	0.29	0
DeepSeek v3.2	−0.13	74	0.78	0.29	2.53	3.42	65	19.3	0.5	20	7	5.28	23
GPT-5.4 Nano	−0.25	74	0.72	0.40	2.47	3.95	65	13.2	0.7	29	9	3.39	17
EconomyAgent	−0.27	172	0.92	0.15	1.90	3.40	40	34.2	0.0	22	0	3.85	1
Claude Haiku 4.5	−0.47	74	0.69	0.33	1.84	3.31	57	11.4	0.8	24	21	0.23	12
Gemini 2.5 Flash Lite	−0.67	71	0.25	0.26	1.93	6.15	56	23.8	1.4	16	15	3.99	25

Table 7: Behavioural metrics per policy in the full 242-game earlier evaluation, with the definitions of Appendix B.3 and the columns of Table 4, sorted by margin score within this tier. Games counts seat appearances; code agents can hold two seats in one game. Spirals are given per game because game counts differ. Sonnet 4.5 has only 14 seat appearances, so its values are uncertain.

opponent, including all three code agents. Gemini 3.1 Flash Lite maintains positive records against all models except Gemini 3 Flash. By TrueSkill on this 98-game slice, TrackerAgent ranks above six of seven LLMs, while DeepSeek v3.2 only beats weaker LLMs.



Figure 16: Auction and TC metrics of the earlier evaluation (98 canonical games, 7 LLMs and 3 code agents): auction participation, win rates, overpay and overbid frequencies, TC counter rate and challenger and target win rates, bluff size in coins, tie frequency, token use per call, truncated answers, and gross coins paid per completed quartet.

Figure 17 shows positional analysis, economic trajectories, and phase-dependent behaviour. The positional win-rate heatmap (left) shows no clear seat advantage for the stronger models in this sample. Money trajectories (centre) show Gemini 3 Flash maintaining higher liquidity throughout games, while Gemini 2.5 Flash Lite's cash reserves deplete rapidly under overbidding penalties. Code agents show stable money trajectories under their deterministic rules. Wealth, unlike score,

can fluctuate; we report the per-turn mean only over games still running at that turn, so late-turn means are biased toward longer-running, more liquid games. 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.

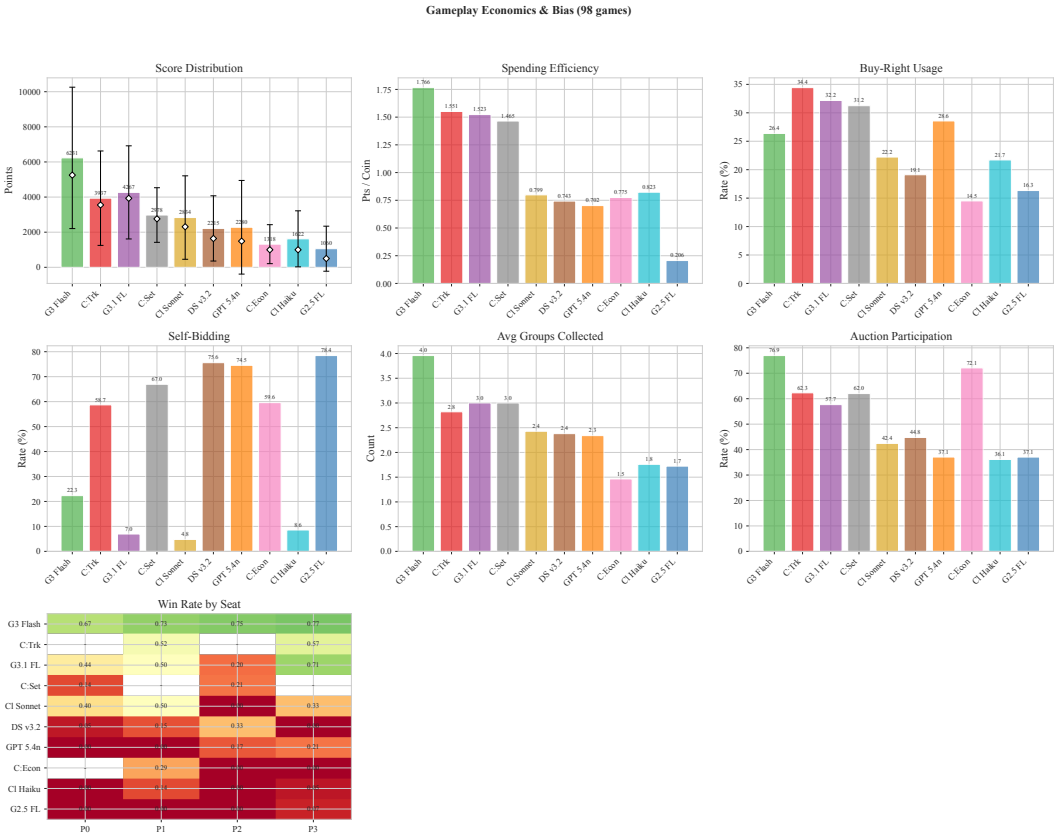


Figure 17: Position and resource use in the earlier evaluation (98 canonical games, 7 LLMs and 3 code agents): win rates by seat (games from each seat as denominator), mean money over turns among running games, accept rate by offer size (TCs received as denominator), exploitation metrics, bid-to-wealth ratio, and TC initiation by donkey phase.

Token usage per game varies roughly 20×: Gemini 3 Flash generates about 275,000 completion tokens per game, while Gemini 3.1 Flash Lite uses about 14,800; both rank at the top of the earlier evaluation.

B.4 Code-agent robustness and self-bidding spirals

Figure 23 and Table 8 report the composition robustness, economic profile and spiral outcomes of the earlier evaluation.

B.5 Bid-jump distribution

The engine requires each new bid to exceed the standing high by at least 10 coins and to be a multiple of 10 (the smallest money-card denomination), but imposes no upper cap. Across the 98-game canonical slice of the earlier evaluation (3,920 canonical auctions, 23,117 positive bid intents), 78.6% of bids hug the minimum +10 step, but 13.4% jump by $\geq +50$, including 5.0% that jump by $\geq +100$ and a long tail reaching a single +1600 intent. Table 9 gives the pooled distribution; Table 10 breaks it down by agent.

The split is sharp. Two of the three Python baselines (EconomyAgent, SetRaceAgent) always play the +10 minimum; TrackerAgent occasionally jumps (5.7% of its bids $\geq +50$). Among LLMs,

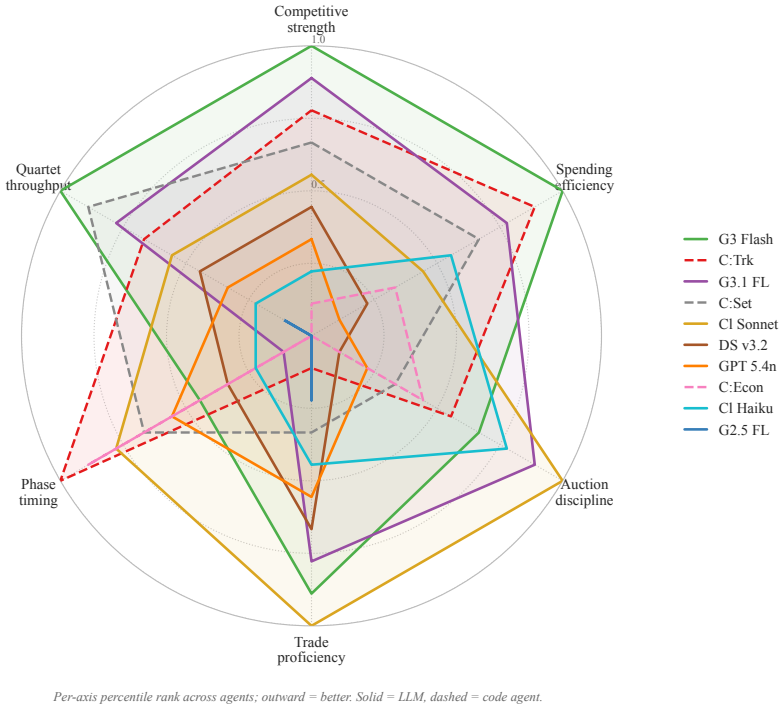


Figure 18: **Strategic-profile radar** on six separately computed axes (98 canonical games, all 10 agents). Each axis is percentile-normalised across the ten agents, so outward means *better*: competitive strength (TrueSkill μ), spending efficiency (points per coin), auction discipline ($1 - \frac{1}{2}$(overbid + self-bid)), TC proficiency (TC challenger win rate), phase timing ($\log$ late/early bid-aggressiveness ratio), and quartet throughput (mean quartets per game). Solid = LLM, dashed = code agent.

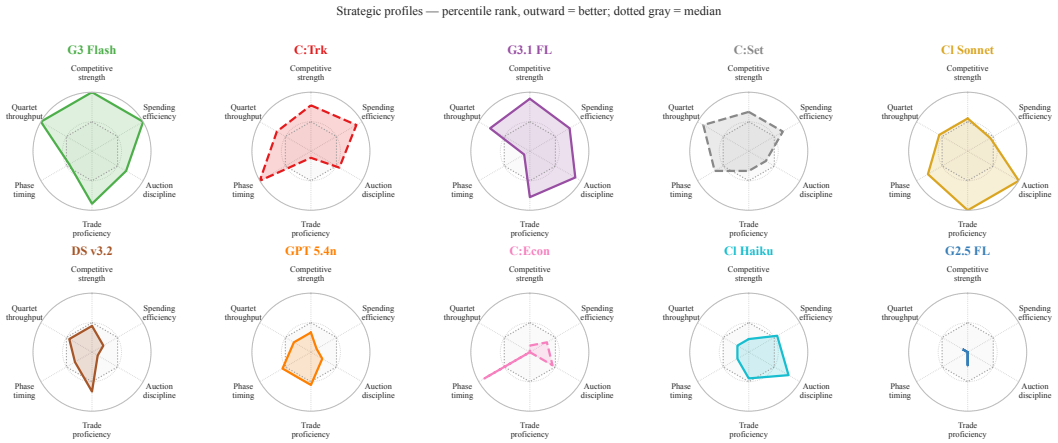


Figure 19: Strategic-profile radar per agent (98 canonical games). Same six axes as Figure 18; each agent plotted against the cross-agent median (dotted grey). The small-multiples view makes per-agent shape asymmetries easier to read than the overlaid version.

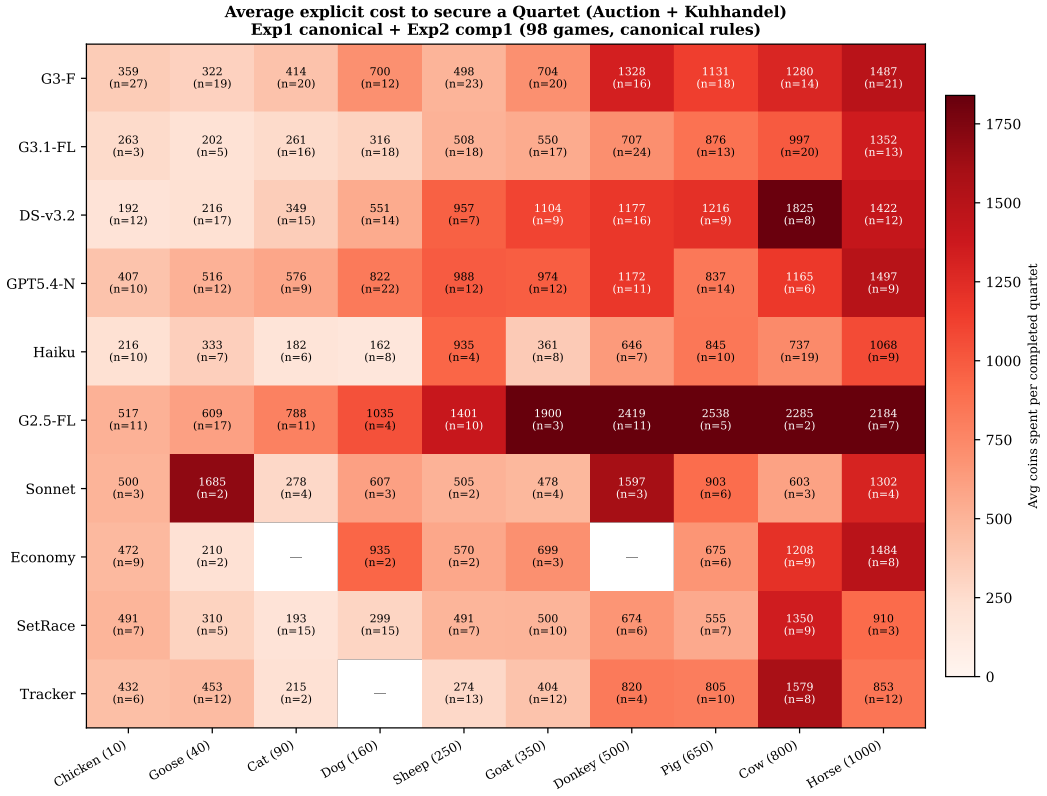


Figure 20: **Gross-outflow cost per quartet** per animal (98 canonical games). Rows: 7 LLMs then 3 code agents; columns: 10 animal types in ascending value order. Cell value: total coins spent by that agent on that animal divided by quartets completed; n: quartets completed (cells with n=0 are blanked, since the ratio is undefined). This is a first-order view; it does not account for TC inflows or the multiplicative scoring effect (see the net-ledger analysis in Figure 23b).

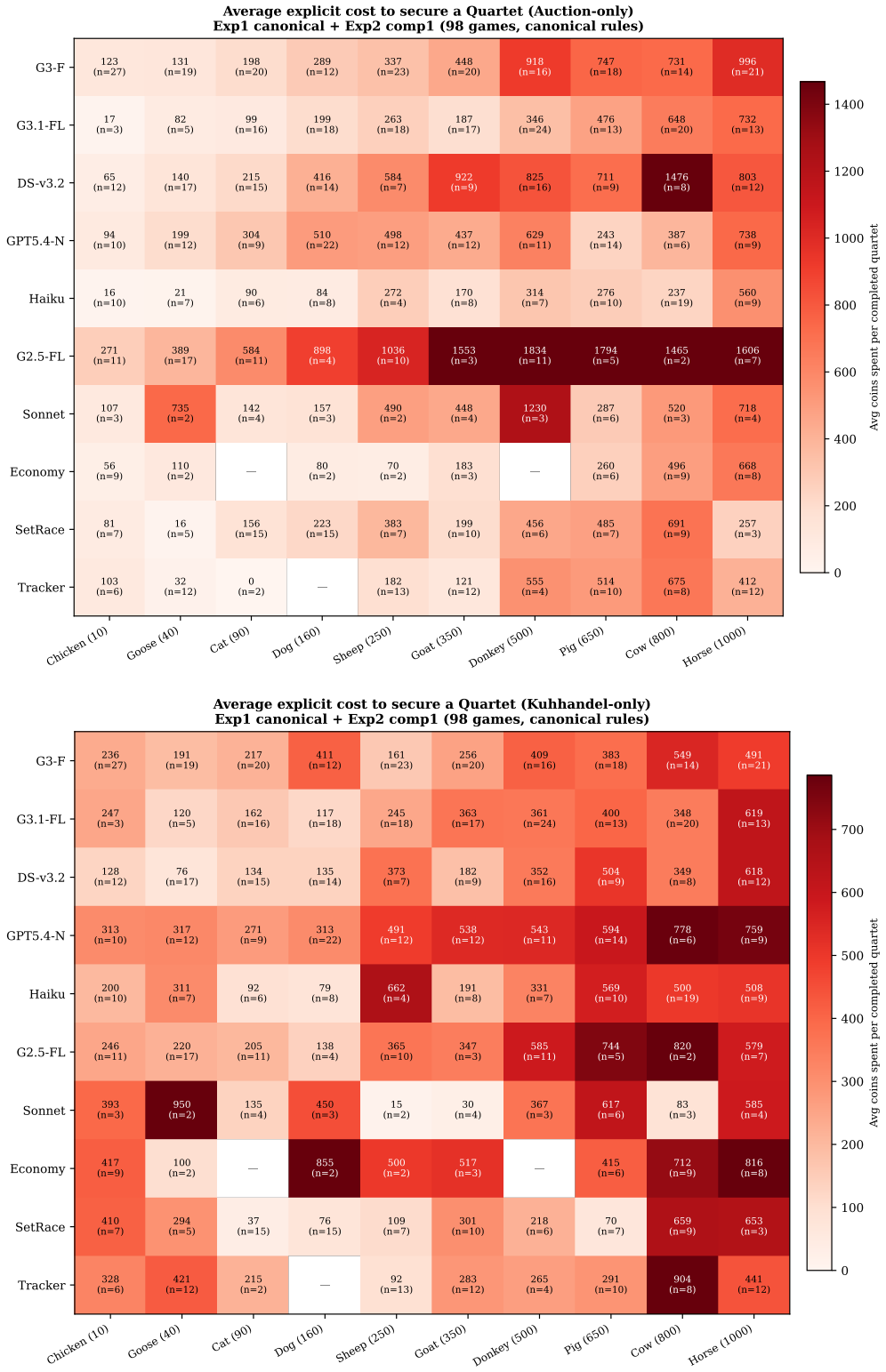


Figure 21: Explicit cost decomposition per animal (98 canonical games). **Top:** auction-only final prices paid by the winner. **Bottom:** TC-only winner's last-round offer value. Auction costs dominate for high-value animals; TC costs concentrate on mid-tier animals.

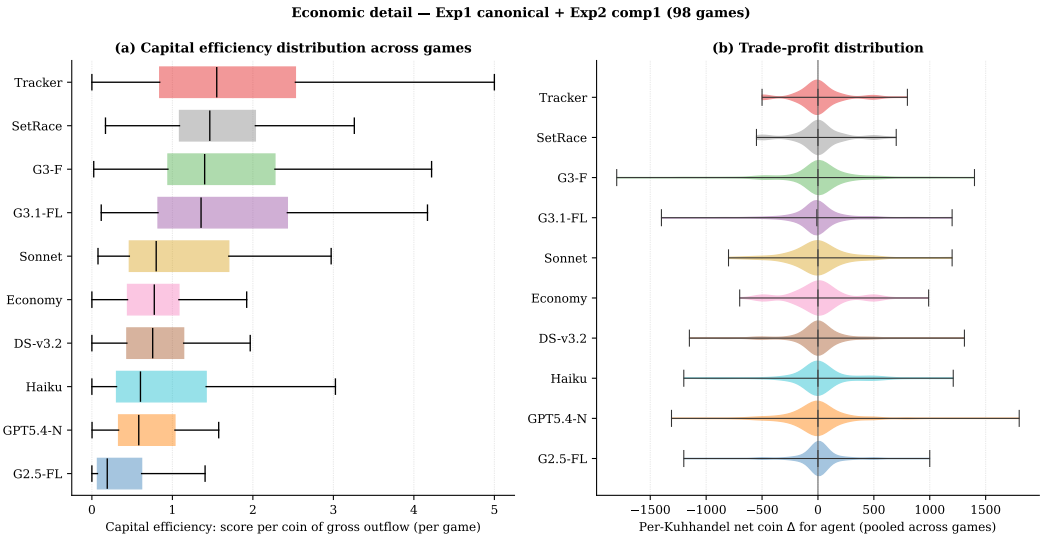


Figure 22: **Economic detail** (98 canonical games). **(a)** Per-game capital efficiency distribution, η = score/gross outflow. **(b)** Pooled per-TC net coin Δ per agent. Mass right of zero = agent pockets money on average; mass left = agent pays to win animals in TCs. The scatter in Figure 23b aggregates these two marginals into per-agent medians with IQR.

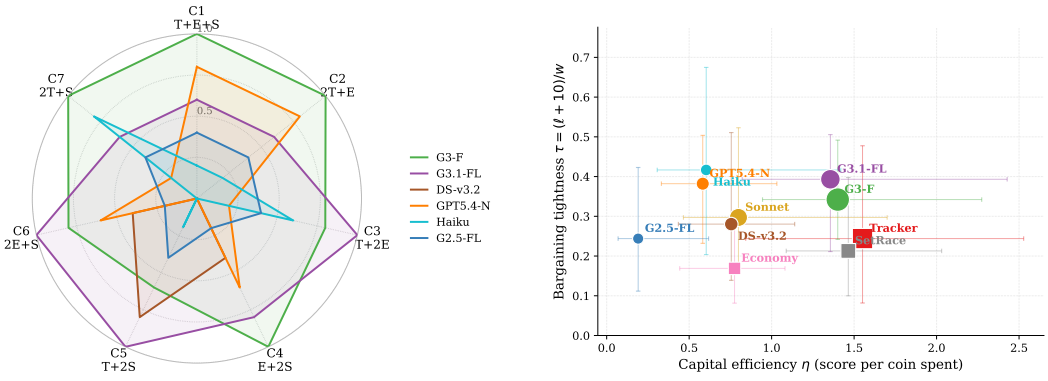


Figure 23: **(a) LLM robustness across 7 code-agent compositions** (168 games of one LLM against three code-agent seats, 4 games per (LLM, composition) cell; Sonnet 4.5's four games against the first composition are omitted because it did not play the others). Axes C1–C7 are seven different mixtures of the three code agents, with repeated seats allowed; 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. **(b) Economic profile** (98 canonical games). x: capital efficiency η, y: TC tightness τ (definitions in Appendix B.3). Median across games, IQR as bars; circles = LLM, squares = code agent; marker size $\propto$ win rate.

Agent	NEED	EXTRACT	DENY	WASTE	LOST	Total	%WASTE
TrackerAgent	39	6	1	0	0	46	0.0%
EconomyAgent	42	43	30	2	1	118	1.7%
Sonnet 4.5	0	2	1	0	1	4	0.0%
Haiku 4.5	4	1	3	1	4	13	7.7%
Gemini 3 Flash	24	10	20	7	4	65	10.8%
Gemini 3.1 Flash Lite	3	5	5	6	13	32	18.8%
GPT-5.4 Nano	86	18	24	31	8	167	18.6%
DeepSeek v3.2	114	37	38	60	7	256	23.4%
SetRaceAgent	96	11	0	35	2	144	24.3%
Gemini 2.5 Flash Lite	58	11	48	46	9	172	26.7%

Table 8: Self-bidding spirals per agent, across the 98-game canonical slice of the earlier evaluation. A *spiral* is ≥ 3 consecutive call-rounds in which one bidder is the only positive bidder and raises their own bid each round. Each spiral is assigned one of five mutually exclusive outcomes from (i) the auctioneer's settlement decision (accept the high bid, or exercise buy-right and keep the card) and (ii) each player's holdings of the contested animal when the auction began. **NEED**: the auctioneer accepted the bid (bidder got the card) and the bidder already held ≥ 1 of the animal; raising aggressively makes buy-right too costly for the auctioneer, and the bidder keeps building toward a quartet. **EXTRACT**: the auctioneer exercised buy-right (auctioneer paid the bidder and kept the card); the bidder pushed the price high enough that paying out was cheaper than selling. **DENY**: the auctioneer accepted the bid with the bidder holding 0, but the auctioneer or another opponent held ≥ 1; the bidder denies that player's progress toward a quartet. **WASTE**: the auctioneer accepted the bid when no player held any of the animal, so there is no observable rational target in the holdings (includes pure overpays plus failed extract or speculative-collection attempts). **LOST**: the bidder did not win the auction. Rows are sorted by %WASTE.

Sonnet 4.5 leaps the most (48.9% of its bids $\geq +50$, but only n_{bids}=229), followed by Gemini 3 Flash and Gemini 3.1 Flash Lite ($\approx 38\%$). Haiku is moderate (23.5%); Gemini 2.5 Flash Lite, GPT-5.4 Nano and DeepSeek v3.2 are conservative (8–17%). The two strongest earlier models also make large jumps often; across the seven models of the full evaluation the association with rank is not significant (ρ=0.54, p=0.22). Large jumps can raise the price of buy-right or price an opponent out of a card it needs; Table 8 reports the corresponding spiral outcomes.

Jump above standing high	Bids	%
+10	18,176	78.6
+20	934	4.0
+30 to +40	788	3.4
+50	823	3.6
+60 to +90	606	2.6
+100	705	3.0
+110 to +190	427	1.8
+200	121	0.5
$\geq +210$	386	1.7
Total	23,117	100.0

Table 9: Pooled bid-jump distribution. +10 is the engine-enforced minimum increment; the long tail (up to +1600) is permitted by the rules and is used in practice.

Agent	n_{bids}	%=+10	%≥+50	%≥+100	max
Claude Sonnet 4.5	229	37.1	48.9	26.6	1210
Gemini 3 Flash	2,131	41.5	38.6	18.7	1600
Gemini 3.1 Flash Lite	1,590	43.5	38.4	21.4	1000
Claude Haiku 4.5	908	64.8	23.5	11.5	1000
Gemini 2.5 Flash Lite	3,562	78.4	16.6	11.5	1000
GPT-5.4 Nano	3,156	77.7	9.3	4.0	780
DeepSeek v3.2	4,195	86.0	8.1	4.2	1000
TrackerAgent	2,080	86.9	5.7	1.2	690
EconomyAgent	2,348	100.0	0.0	0.0	10
SetRaceAgent	2,918	100.0	0.0	0.0	10

Table 10: Per-agent jump frequency, sorted by %≥+50. The Sonnet 4.5 row rests on 229 bids from 14 seat appearances and carries wide uncertainty.

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:

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 (TC).
4. **Memory**: the full history of events visible to the player and the player's private notes (see below). In the earlier cost-efficient evaluation (Appendix B) the memory held only the most recent $\sim$10 events plus the notes.

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 free-text analysis in an `<analysis>` tag followed by a JSON object in a `<move>` tag; the JSON object carries a one- or two-sentence `rationale` and the action fields. Seven decision types arise; monospaced labels are output values or JSON fields, and `kuhhandel` is the implementation's name for a trade challenge:

1. **Turn choice**: {`auction, kuhhandel`}:select whether to draw a card for auction or initiate a TC 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.
4. **TC 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. **TC defense**: {`accept, counter(cards)`}:accept the face-down offer (take money sight-unseen, surrender animals) or counter with specific money cards from hand.
6. **TC tie retry**: {`offer_cards`}:after a tie, submit a new offer using specific cards. After three ties, the initiator wins by default.
7. **Notes update**: free-text notes on opponents and the player's own plan, returned without tags.

Error handling by action type. The framework validates all actions and applies action-specific recovery procedures. Table 11 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).

Notes mechanism. Each agent keeps private notes s^i. After new events, the framework issues an additional call to the *same model* with the current notes, the player's money total and animal holdings, the events since the last successful update and the player's own moves since then, and asks for durable inferences only: each opponent's tendencies, bluff and call patterns, likely quartet targets, and the player's own plan (fewer than 180 tokens requested, 240 at most; stored up to 1,300 characters). The update is all-or-nothing: a refused, empty or truncated reply leaves the notes unchanged and keeps the events pending for the next update. The notes appear in every later decision prompt under the heading "YOUR NOTES", next to the full event history. The model's own earlier analyses are not fed back.

League configuration. Table 12 lists the settings shared by all LLM policies in the league.

Action Type	Error Case	Handling
All actions	Unparseable JSON	Multi-stage fallback on the last complete `<move>` block: (1) direct parse, (2) regex extraction of the last JSON object, (3) manual key-value extraction with type coercion
All actions	Move missing, unparseable or truncated	Up to two move-only retries at low effort; if both fail, a default legal action is played and logged
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 (LLMs and, in the league, code agents)
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
TC 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
TC initiation	Offer cards not in hand	One retry with error feedback showing available cards; fallback to greedy card selection
TC 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 11: 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.

Setting	Value
Players, deck	4 players, 10 animal types, k=4
Auction mode	canonical (simultaneous-intent bidding)
System prompt	neutral rules + two-line goal statement (Appendix D)
Temperature	1.0
Output limit	32,768 tokens per call
Reasoning effort	low for all models; high in addition for Fable 5.1, Opus 5.5, Sonnet 5
Memory	full observed event history + private notes (no fed-back analyses)
Answer format	`<analysis>` free text, then `<move>` JSON with `rationale`
Retry	up to two move-only retries at low effort, then default legal action
Routing	LiteLLM; provider APIs; open-weight models on dedicated endpoints
Logging	per call: model id returned, effort sent and accepted, tokens, cost, latency
Schedule	182 games, 13 deck seeds, 52 games per policy, 12 meetings per pair

Table 12: League configuration shared by all 11 LLM policies.

D AGENT PROMPTS

The templates below reproduce the game-specific prompt text of the benchmark implementation. Template variables (in angle brackets) are filled dynamically from game state. In the league, every action prompt ends with the answer-format block of Section D.2 instead of the JSON-only response line shown in several templates. Long lines are wrapped for print.

D.1 SYSTEM PROMPT

Every LLM call in the league uses the system prompt below: a header, a two-line instruction to maximize expected score, and the neutral game-rules block. The instruction names the goal without prescribing how to achieve it; no strategy hints, examples or tactical advice are added. The benchmark supports other prompt variants, which are not evaluated here. The earlier cost-efficient evaluation (Appendix B) used the same instruction under the name "Kuhhandel Master".

```
KUHHANDEL - RULES

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

GOAL: Complete quartets (4 cards of same animal). Score = (sum of
  complete quartet values) x (number of quartets).
Example: 1 donkey quartet (500) + 1 horse quartet (1000) = (500+1000) x 2
  quartets = 3000 points.
Example: 1 horse quartet alone = 1000 x 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.
    DONKEY: When a Donkey is auctioned, EVERY player first receives money
      from the bank
    (50 for the 1st Donkey, then 100, 200, 500 for the 2nd/3rd/4th), then
      it is auctioned normally.
    TIES: if two bids are equal, the winner is decided by a random
      priority order drawn
    for that auction (not by turn order).

[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 Answer Format

Every action prompt below ends with the following instruction, in which the second line of the <move> block holds that action's JSON fields. Where the templates below show `Respond with JSON: {"reasoning": ..., <fields>}`, the league prompt ends with this block instead (shown for the turn choice). The framework parses the last complete <move> block.

```
Write your answer in exactly this format and nothing else:

<analysis>
The decisive facts of the current state, your main options,
the tradeoff between them, and why you choose your move.
Plain text.
</analysis>
<move>
{"rationale": "<one or two sentences>",
 "action": "auction" or "kuhhandel"}
</move>
```

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>
```

In the league, the full event history and the notes are sent as separate blocks between the system prompt and the observation, under the headings `GAME HISTORY (everything you have observed):` and `YOUR NOTES (your read on opponents' tendencies and your plan):`; the recent-events and notes sections shown above belong to the earlier evaluation. 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. A first overbid reveals the bidder's wealth to all opponents and restarts the auction; a second overbid by that player eliminates them 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 TRADE CHALLENGE 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 Trade Challenge 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 Trade Challenge 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 NOTES UPDATE

After new events, the framework sends the following prompt as a single user message (no system prompt) to the same model. The opponent lines list the player's three opponents; the first carries the example text. The reply replaces the notes only if it is non-empty, not refused and not truncated. Lines are wrapped for print.

```
You are P<id> (Player <id>). Update your private notes. You WILL
still see the full event log next turn, so include ONLY durable
INFERENCES the log does not state: each opponent's behavioral
tendencies, bluff/call patterns, likely quartet pursuits,
predicted near-term moves, and your own strategy for the rest of
the game. Do NOT restate observable state, card counts, coin
holdings, or auction history (you can already see those). Keep
notes under 180 tokens; 240 max. Use this terse format:
P<j>: behavioral read / likely target / bluff pattern.
P<k>: ...
P<l>: ...
Plan: current commitment + next likely action.

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

Your position (P<id>): money total <total>; animals held: <animals>

New events:
<events since the last applied update, or "(none)">

Your own moves (P<id>) since your last update:
<own moves, or "(none)">

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

E SCENARIO DATASET AND BEHAVIOURAL AXES

Each scenario record holds a stable identifier, the source game and decision index, the acting player and model, the acting player's information state, the public event history, the legal action set, the logged prompt and response with token usage, the logged action, the final scores and the engine-exact labels. A separate hidden-state record holds the full engine snapshot; the scenario builder and all prompt renderers never read it. Bids are discretised to pass plus the legal multiples of 10 up to the actor's wealth, plus one overbid probe; trade offers are the distinct sub-multisets of the actor's money cards. Every record is reproducible from the trace and the seed, and the pipeline produces byte-identical files on repeated runs. Synthetic records carry a provenance flag, the steering events that created them and their contrast twin.

Table 13 defines the behavioural axes of Figure 3. An axis applies when its trigger holds in the visible state; the score uses the logged action and the recorded outcome. Forced moves (a single legal action) are excluded.

E.1 RECURRING ERRORS IN THE MODELS' OWN WORDS

A reader model (Codex gpt-6-sol) went through 352 sampled decisions and flagged 41 errors, each with the prompt facts that prove it and a better move; all 41 quotes appear verbatim in the model's

Table 13: Behavioural axes. Trigger conditions use only the acting player's visible state.

Axis	Trigger	$+1$	-1
Keep needed card	actor auctions a card it holds 2–3 of	keeps it below its score gain	pays above its score gain, or sells cheaply
Complete own quartet	actor holds 3 of the auctioned card	wins it	lets it go while affordable
Cheap multiplier	actor holds 3 of an animal worth ≤ 160	wins it, even above face value	lets it go while affordable
Broke: auction	actor has no money, deck not empty	auctions	starts a trade with an empty offer
Milk needy auctioneer	auctioneer holds 2–3, actor 0–1	raises the price	wins it far above its value to the actor
Pressure a completer	another bidder holds 3	raises the price or wins at $\leq$ value	wins far above value
Block a completer	actor auctions; high bidder holds 3	buy-right	sells cheaply ($< 50\%$ of value)
Known opponent wealth	opponent money exactly known	offers just above it	pays far above it, or bids below it
Endgame cash	≤ 2 cards left; a player holds 3 of the contested card	wins the decisive contest	loses it while able to pay

text. They fall into a few families: miscounted cards or hidden values treated as known, incomplete sets counted as scoring, bids that ignore the no-change rule, raising one's own standing bid, underrating an opponent's gain from a second quartet, and expecting more cards to move in a trade than the rules allow. An example of the last family: "winning this trade completes a 650-point quartet" (GPT-6 Sol, holding one pig against an opponent's three), where the trade moves one pig. The rules text states the two-card case (both players hold two) and implies the single-card case without spelling it out. A weaker reader (Haiku 4.5) was far less reliable: 10 of its 24 flagged quotes were not verbatim, and it agreed with the stronger reader on only 63% of quality ratings. We therefore use LLM readers to discover situations and score behaviour from the logs.

E.2 Scenario Suite

The mined scenarios (89,638 from the earlier evaluation after deduplication, plus the league decisions) carry engine-exact labels such as affordability, the least-overpay payment, legal trade targets, trade sizes and exact score changes. To cover rare situations, we also instantiate each human strategy pattern many times in legal engine games: code agents play normally and only the few events the pattern needs are steered, so the public history explains the state. Every instance has a contrast twin in which the mechanism is switched off (the auctioneer is broke, or a wealth reveal is stale). Synthetic items are marked as such, use a clarified rules text for trade sizes, and never replace the real decisions. The suite renders each scenario as the in-game prompt, as a clean prompt without history, as structured JSON, with a public-state summary and with exact arithmetic helpers, which separates parsing, memory and arithmetic from the strategic choice (Table 13).