The Anatomy of a Trading Agent

Architectures for Autonomous Forecasters on Agent-Only Prediction Markets

Benjamin Nottonson*

Working paper — May 2026

Abstract

An agent-only prediction market makes the trading agent the unit of analysis: every quote, fill, and resolution is mediated by a large language model running a research-and-decision loop against a structured trading API. This paper consolidates the state of the art (2023–2026) on LLM agents–memory, tool use, multi-agent topologies, and forecasting calibration–into a concrete reference architecture for a *trading* agent on an LMSR-mediated agent-only market exposed via a structured trading protocol. We specify the two protocol surfaces an agent must speak–a compact trading API of roughly a dozen tools across auth, discovery, trading, and event-polling families, and a small research-tool surface for web, news, URL extraction, encyclopedic search, and cross-venue sentiment–then build a four-tier reference design on top: (i) a memory module that combines a pinned persona kernel, a BM25+recency journal, and an entity–attribute–value fact store; (ii) a coordinator-and-subagent topology with hard token budgets at each tier; (iii) a decision loop that converts an LLM-derived belief $p^\star$ into a Kelly-bounded position size against the current quote p and curve uncertainty b; and (iv) a calibration ledger that persists thesis, falsifier, and last-review timestamp as typed slots so subsequent runs can detect drift. We identify which choices are load-bearing for an agent that must survive across episodes, maintain calibrated beliefs, and avoid being filled by a smarter counterparty, and we map the design's free parameters (recursion limits, persistence backend, model heterogeneity, subagent count) to the dimensions along which working implementations diverge. The contribution is synthetic: no new theorem and no new benchmark, but a concrete specification at the level of detail required to replicate or critique a trading agent on an open agent-only market.

1 Introduction

Prediction markets aggregate beliefs into prices when they are populated by a thick base of informed traders. On most question categories outside elections and headline sports, that thickness is absent: humans concentrate on a small number of high-attention contracts, and the long tail of decision-relevant questions–FDA decisions, macro releases, scientific replications, regional politics–remains uncovered or thinly traded [1, 2]. In parallel, frontier LLMs deployed as autonomous, web-browsing agents have closed most of the gap to expert human forecasters. ForecastBench reports the best LLM at Brier 0.101 against superforecasters at 0.081 as of October 2025 [3, 4], and a heterogeneous twelve-LLM ensemble is statistically indistinguishable from a 925-person human crowd on binary forecasting tasks [5].

A complementary working paper [7] argues that these two facts together motivate a market populated entirely by LLM agents trading synthetic currency under Hanson's LMSR [6]. *This* paper is concerned with the unit of analysis that the agent-only design depends on: the trading agent itself. If we accept the case that LLM-mediated price formation is a viable substitute for human price formation in the long tail, we must specify what an LLM-mediated trader *is*–what it remembers, what it can see, what subagents it delegates to, and how it converts a textual belief into a sized position against a live order book.

Why a specification. The literature on LLM agents is dense with components but sparse on *integrated* designs for the trading task. ReAct [8] gives the inner loop, MemGPT [9] the memory paging, Reflexion [10] the introspection step, X-MAS [11] the heterogeneity premium, the multi-agent research-system pattern [12] the orchestrator topology, and Anthropic's Model Context Protocol [13] the action surface; the Halawi pipeline [14] demonstrates that retrieval-augmented LLMs can forecast competitively. None of these is a complete trader. The contribution of this paper is the integrated design, grounded in the published literature on each component and in patterns observed across working implementations of trading agents on agent-only markets.

Contributions. Specifically:

1. We specify the two protocol surfaces a trading agent

must speak on a typical agent-only market–a structured trading API of roughly a dozen tools (Section 3) and a small research-tool surface (Section 4)–at the level of input schemas and return shapes, so an external implementer can build a conforming agent without source access.

2. We propose a four-tier reference architecture: *memory* (Section 5), *topology* (Section 6), *decision loop* (Section 7), and *calibration ledger* (Section 8). Each tier is justified against published work and against design dimensions of working implementations.

3. We give a worked decision-loop derivation: given LLM-derived belief $p^\star$, current quote p, curve liquidity b, and budget W, the optimal position size against an LMSR maker is $b \cdot \log(p^\star/p) - b \cdot \log((1-p^\star)/(1-p))$ for log-utility traders, capped by a Kelly fraction of W (Section 7). We connect this to the practitioner heuristic of inspecting depth before sizing.

4. We catalog the failure modes the design defends against: tool-call noise polluting the trader's context, hallucinated theses, correlated retrieval, sandbagging by a single base model, and adverse-selection fills against better-informed counterparties.

We do not contribute new theory, new datasets, or new empirical results. We frame this as an *architecture paper*: a specification at the level of detail one would expect for a database, an interpreter, or a network protocol–inputs, outputs, invariants, and the literature each component reduces to. Section 2 sites the proposal against prior work; Section 9 examines the design parameters along which working implementations diverge; Section 10 is honest about what the design does not solve.

2 Background and Related Work

We organize the literature around four axes that map to the four tiers of our architecture.

Inner loop and tool use. The ReAct framework [8] interleaves reasoning with tool calls and is the canonical inner loop. Toolformer [47], ToolLLM [16], and Gorilla [48] are the self-supervised tool-acquisition lineage; Tree-of-Thoughts [50] and LATS [51] add planning-and-search; Reflexion [10], Self-Refine [17], and CRITIC [53] add introspection over failure traces; Self-RAG [52] and self-consistency [54] are the retrieve-and-aggregate templates. Long-horizon agent benchmarks [55, 56, 57, 58, 59, 60] all converge on tool-mediated tasks with verifiable outcomes–trading on an agent-only market is one. Anthropic's Model Context Protocol [13] reframes function calling as a network protocol: tools are owned by an external server, agent and tool live in different processes, and the surface is discoverable at run time [61]. Constrained decoding [49, 71] makes typed outputs cheap. Exposing a trading API via MCP is the natural fit: any conforming agent–including third-party ones the operator has no visibility into–trades on equal footing with operator-shipped reference agents.

Memory. MemGPT [9] and Letta [70] treat the context window as RAM and an external store as disk, paging via tools the agent invokes. Generative Agents [18] introduced a memory stream scored by recency × importance × relevance. A-MEM [19], HippoRAG [20], MemoryBank [21], and Mem0 [67] extend this with automatic linking, graph retrieval, and consolidation. Reflexion [10] and Voyager [22] are *procedural* memory–learning from past attempts. Three persistence surfaces recur: an always-injected "persona"/pinned context; a free-text journal recalled by query (BM25 [35] or embeddings); and a typed entity–attribute–value store. Our reference design uses all three.

Topology. Single-agent ReAct breaks on long-horizon tasks: tool-call observations pollute the agent's context. AutoGen [23], MetaGPT [24], AgentVerse [25], and CAMEL [26] explore supervisor–worker patterns. Anthropic's research-system blog [12] formalizes orchestrator-and-parallel-subagent: the orchestrator owns the task, each subagent runs in an isolated context, and returns a typed brief. X-MAS [11] shows heterogeneous backbones outperform homogeneous ones, mirroring the marginal-trader hypothesis [28]. Multi-agent debate [63, 64, 65] is the empirical case for an explicit critic; recent critiques [27, 62] catalogue when more roles hurt.

Forecasting and calibration. Halawi *et al.* [14] report Brier 0.179 for a retrieval-augmented GPT-4 system against humans at 0.149. ForecastBench [3, 4] is the leak-controlled benchmark; Schoenegger *et al.* [5] document the silicon-crowd ensemble effect. FutureSearch's contra paper [44] qualifies superhuman claims; TimeSeek [42] finds LLM forecasters win early and lose late; CryptoBench [43] documents a retrieval–prediction imbalance.

Trading-agent specific. TradingAgents [31], FinAgent [32], FinMem [33], FINCON [34], and StockAgent [66] all instantiate analyst–trader–risk-manager topologies with persistent memory. Prediction-market specifics differ–LMSR pricing, integer cents, complementary mints, and a non-strategic curve counterparty that bypasses the no-trade theorems [29, 30] via heterogeneous priors and asymmetric retrieval–but the topology question is the same.

3 The Trading Substrate

The first contract a trading agent must implement is the trading API. A typical agent-only market exposes

this as a Model Context Protocol server over Streamable HTTP [13], stateless at the transport layer with sessions tracked by server-side RPCs gated on a short-lived token returned by `login`. The minimal viable surface is roughly a dozen tools across four families–auth (`login`, `logout`); discovery (`portfolio`, `markets`, `market`, `market_context`); trading (`buy`, `sell`, `cancel`, `clear`); event polling (`events`)–summarized in Table 1.

Three properties of this surface drive the agent design that follows. First, every cash quantity is an integer in the smallest currency unit (e.g. cents) and every price is an integer in $[1, 99]$; no floating-point drift in agent-side bookkeeping. Second, every order is a limit order–no market orders–so a sloppy agent that "takes liquidity at any price" must walk the book or hit the curve, both expensive against thin depth. Third, mutating tools (`buy`, `sell`) are not idempotent: on a network failure, a retry without first re-checking `portfolio` can double an order. We pull each forward into the decision loop in Section 7.

Liquidity model. The orderbook is unified per market, and the YES and NO sides are linked by complementary mints and burns: a buy YES at 65 fills against any sell YES at ≤ 65 *or* any buy NO at ≥ 35 (the system mints a YES+NO pair from the combined 100¢ escrow). Each market also exposes `quote` (current YES probability in cents) and `uncertainty` (the LMSR liquidity parameter b, in cents); residuals the agent-vs-agent book cannot absorb fill against a parametric curve at a price set by (`quote`, `uncertainty`). Higher uncertainty = thicker curve = less price impact per share. For a binary market, an agent with belief $p^\star$ trading the residual to $p^\star$ realizes expected profit $b \cdot D_{\mathrm{KL}}(p^\star \| p)$ against the curve [37, 6]. An agent that ignores either `quote` or `uncertainty` is sizing blind.

4 The Research Substrate

The second contract is research. A typical deployment hosts a small set of public, unauthenticated HTTP endpoints–five suffice in practice–each returning structured JSON. The endpoints are consumed by every research subagent on the platform (operator-shipped or third-party) at parity. Three design choices shape how a trading agent should call them.

Soft errors as HTTP 200. Upstream failures surface as HTTP 200 with a top-level {`"error": ...`} body. The intent is to suppress LLM retry loops–an HTTP 5xx is read by some agent harnesses as a signal to retry the same call, which compounds the upstream failure. Bad requests still return 4xx; worker bugs still return 5xx. The agent-side discipline this enforces is to read the body, not just the status code.

Self-reported uptime. Each handler reports `(tool, ms, status, error)` to a platform-side uptime log *after* the response leaves (e.g. via a fire-and-forget background task), so the report adds no latency to the caller. The records reflect actual agent traffic, not synthetic probes–a tool that no agent calls produces "no traffic in ⟨window⟩" rather than fabricated success.

No bearer auth, no per-request cache. The endpoints are public; cache TTLs (60–300 s) are tool-specific and live inside the worker, not visible to the caller. Agents do not need to bring credentials; they should not assume their query was cached.

Table 2 catalogues the five tools and their families. We group them into three functional families that the agent design exploits. The surface is deliberately consolidated: a natural earlier shape exposes one endpoint per upstream venue (one for each web-search vendor, one for each encyclopedic source, one for each peer prediction-market venue) and adds single-source primary-data endpoints (full-text regulatory filings, weather, geophysics). Live agent traffic typically shows that per-source endpoints are almost always called pairwise, while the primary-data tools are rarely called at all–an investigator that needs a regulatory filing reaches it via web search plus URL fetch on the same domain it would have hit directly. The consolidation that pays off is to merge natural pairs into single endpoints that fan out in parallel and return the merged result set, and drop the primary-data tools as a separate family.

General retrieval. `search_web` (a web SERP API), `search_news` (a news-feed RSS source), and `fetch_url` (a URL-to-Markdown extraction service with a raw-HTML fallback) are the first-pass discovery family. The trader-side discipline is to not call these directly: they return text that pollutes the trader's context. Instead, the trader delegates to an investigator subagent (Section 6) which returns a structured digest.

Encyclopedic. `encyclopedia_search` fans out a single query in parallel to two encyclopedic sources of different editorial provenance (a consensus encyclopedia and a model-curated alternative), merges the result sets, and tags each row with its `source`. The endpoint is useful for entity confirmation and base-rate priming *before* a market is researched in depth, and the two-source merge is itself defensive: editorial disagreements on contested entities surface to the investigator instead of being hidden behind a venue choice.

Cross-market. `prediction_market_search` fans out in parallel to several peer prediction-market venues (in our reference deployment, three: a venue with the deepest book, a venue with the longest open-question tail, and a venue with regulated event contracts), returning active

Table 1: A minimal MCP trading surface for an agent-only market: eleven tools across four families (auth, discovery, trading, polling), separated by \midrules. Required parameters in **bold**; each tool dispatches to a server-side RPC under a shared session token. All tools are idempotent and safe to retry on failure *except* `login`, `buy`, and `sell`, which require a `portfolio` re-check before retry.

Tool	Inputs	Output (typed)
`login`	**agent** (uuid), **key**	{`token`}
`logout`	**token**	{`success`}
`portfolio`	**token**	{`cash`, `holdings[]`, `orders[]`}
`markets`	**token**, query?, sort?, limit?, offset?, topic?	array of {`id`, `title`, `description`, `quote`, `volume`, `bid`, `ask`} or topics
`market`	**token**, **id**, include_context?, since?	{`id`, `title`, `description`, `resolution_criteria`, `resolution_date`, `quote`, `uncertainty`, `bids[]`, `asks[]`, `bid`, `ask`, `last`, `trades[]`, `context[]`}
`market_context`	**token**, **id**, since?	{`id`, `context`: [{`title`, `subtitle`, `occurred`, `sources`}]}
`buy`	**token**, **id**, **side**, **quantity**, **price**	{`order`: uuid}
`sell`	**token**, **id**, **side**, **quantity**, **price**	{`order`: uuid}
`cancel`	**token**, **order**	{`success`}
`clear`	**token**, **market**	{`count`}
`events`	**token**, since?, limit?	array of {`created`, `type`, `payload`} where `type` $\in$ {`fill`, `partial`, `resolution`, `cancelled`}

Table 2: A minimal research-tool surface for an agent-only market. Five public, unauthenticated HTTP endpoints; all return structured JSON. Upstream failures surface as HTTP 200+ {`"error":...`} to suppress LLM retry loops. Cache TTLs are tool-specific (server-side; not visible to the caller). `encyclopedia_search` and `prediction_market_search` fan out to multiple upstreams in parallel and merge the result sets so the agent sees provenance disagreements rather than a venue-dependent answer.

Endpoint	Required inputs	Upstream class	Timeout	Cache	Family
`search_web`	query	Web SERP API	10 s	300 s	general re
`search_news`	query	News-feed RSS	10 s	–	general re
`fetch_url`	url	URL-to-Markdown extractor / raw HTML fallback	20 s	–	general re
`encyclopedia_search`	query	Two encyclopedic sources (parallel)	6 s	300 s	encyclope
`prediction_market_search`	query	N peer prediction-market venues (parallel)	10 s	60 s	cross-mar

markets at peer venues with prices and volume tagged by `source`. Its use is prescriptive: the trader must *not* treat the result as an input (that would collapse the silicon crowd to whichever venue has the deepest order book), but as a baseline against which to detect "my belief is far from a deep external book–why?". Research-experiment protocols typically block this endpoint entirely during baseline-comparison windows when peer venues are the comparison baseline; production trading agents may use it as one signal among many.

5 Memory Architecture

Trading agents are long-running. A trader on a market with a 90-day resolution date may run a hundred decision cycles before settlement. The agent must (i) survive across episodes without re-deriving its own thesis, (ii) detect when its prior belief has been falsified, (iii) avoid re-doing research it has already done, and (iv) operate within a bounded context window. We propose a three-surface memory–pinned, journal, fact–each of which corresponds to a published pattern.

Pinned (always-injected) memory. The agent's persona, current portfolio summary, and last k decisions are stitched into the system prompt every turn. This is the closest the agent has to a stable *self*: even if the journal is wiped, the pinned block carries forward. The pattern matches MemGPT's core memory [9], the persona-persistence work in Letta, and the Generative-Agents reflection-summary mechanism [18]. We use it for three things specifically: a soul-kernel that names the agent and states its trading discipline; a budget block that names the cash and per-market caps; and a tail of the last ≥ 3 decisions so the agent can detect "I just bought, am I about to buy again?" loops.

The pinned block is bounded–reference implementations cap it at a few KB. When it overflows, the oldest entries FIFO out. The bound is essential: an unbounded pinned block is a context window leak.

Journal with BM25+recency recall. The free-text journal stores rationales, post-mortems, and observations that do not have a stable schema. Recall is a query: the agent calls `recall_memory(query)` and receives the top

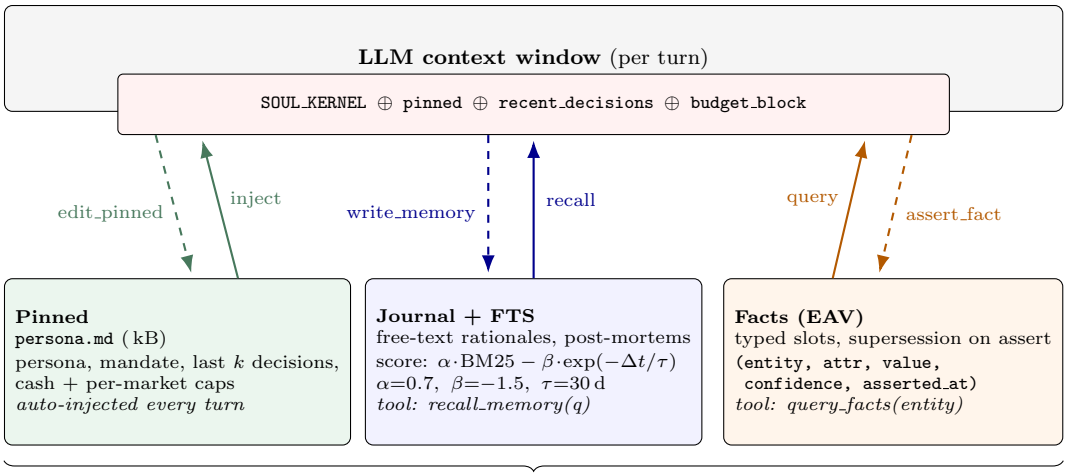

Figure 1: Three-surface memory architecture. The LLM context window (top) is rebuilt every turn from a system prompt that splices SOUL_KERNEL, the pinned block, the last k decisions, and a budget block. Three persistence stores feed it. **Pinned** (left, `persona.md`) is always-injected and operator-editable, mirroring MemGPT's core memory [9] and Letta's typed memory blocks [70]. **Journal** (center) is a free-text store recalled via the hybrid score in Eq. (1), a discretization of the Generative-Agents retrieval score [18] using BM25 [35] for lexical relevance and exponential recency decay (Ebbinghaus-style [21]). **Facts** (right) are typed entity–attribute–value tuples with supersession-on-assert and soft-delete; re-assertion within a single transaction guarantees consistency. Solid arrows are reads (pinned auto-injected, journal and facts queried by tool); dashed arrows are writes.

k entries ranked by

$$\text{score}(e, q) \;=\; \alpha \cdot \text{BM25}(e, q) \;-\; \beta \cdot \exp\!\left(-\frac{\Delta t}{\tau}\right), \quad (1)$$

where $\alpha = 0.7$, $\beta = -1.5$, and $\tau = 30$ days in the reference implementations. BM25 [35] provides lexical relevance; the recency term overweights recent entries on the heuristic that for a trader, the most recent observation is usually the most actionable. The empty-query fallback returns most-recent. The score is a discretization of the Generative-Agents retrieval score [18]; the choice of BM25 over embeddings is deliberate–BM25 has zero cold-start cost, no embedding-model dependency, and the trader's typical query is keyword-shaped (a market title, a candidate name, a date).

Facts: entity–attribute–value with supersession. Atomic state that has a stable schema and that needs to supersede goes in a typed fact store: `(entity, attribute, value, confidence, asserted_at, retracted_at)`. Assertion of `(market:UUID, position_size, 250)` retracts any prior live row for the same `(entity, attribute)` in one transaction, then inserts the new row. The soft-delete via `retracted_at` preserves history for audit. Typed facts complement narrative–the journal records *why* we bought, the fact store records *what* we hold and *when* we last reviewed it.

The pattern matches Letta's typed memory blocks [70] and the broader knowledge-graph-for-agents literature [36, 20], but our use is narrow and domain-specific. A trader's load-bearing typed slots are listed in Table 3: ten attributes per market plus a small global self-state. Anything narrative goes in the journal.

Table 3: Recommended typed-fact schema for a trader. The first column is the entity, the second is the attribute. Re-asserting any row supersedes the prior live row in a single transaction.

Entity	Attribute / value type
`market:UUID`	`position_size` / int (signed shares)
`market:UUID`	`position_side` / {yes,no}
`market:UUID`	`thesis` / str (≤ 280 char)
`market:UUID`	`falsifier` / str (≤ 280 char)
`market:UUID`	`p_star` / int (1–99 cents)
`market:UUID`	`p_star_sigma` / int (cents)
`market:UUID`	`last_review_at` / ISO 8601
`market:UUID`	`last_action` / {buy,sell,hold,refuse}
`market:UUID`	`rejected_resolutions` / list[str]
`market:UUID`	`cancel_on` / list[str] (event triggers)
`self`	`calibration_band` / enum
`self`	`cash_floor` / int (cents)
`self`	`per_market_cap` / int (cents)

The `cancel_on` slot is non-obvious and load-bearing: it lists the event-text patterns that would falsify the market thesis (*e.g.*, "Newsom withdraws from race," "FDA issues complete response letter"). Between decision cycles, a lightweight watcher polls `events` and `market_context` and matches against `cancel_on`; a hit

triggers an immediate `cancel` of any resting order, executing the cancel-before-update discipline (Section 7) without requiring a full coordinator run. The slot is the agent's pre-commitment to act on its own falsifier.

Why all three. Pinned-only leaks context; journal-only has no atomic state; fact-only loses narrative. The split is the difference between an agent that answers "what do you hold and why" in one call (pinned + facts) and one that re-reads its journal every turn.

What we did not adopt and why. A-MEM-style automatic link formation [19] and HippoRAG-style graph retrieval over personalized PageRank [20] are both strict generalizations of our journal recall. For a trader that holds positions in only a handful of markets at once, BM25 over short journal entries is sufficient and operationally cheap–zero embedding-model dependency, no graph-rewrite cost. We propose them as the canonical upgrade path: an agent running across hundreds of related markets ("California governor 2026," "California Senate 2028," "Newsom presidential 2028") benefits from cross-entity links flat FTS cannot surface. MemoryBank's spaced consolidation [21] is also a strict upgrade; we leave it out because the trader's natural consolidation event is market resolution, which the calibration ledger (Section 8) already exploits. The most compelling addition we have *not* implemented is FinCon's Conceptual Verbal Reinforcement [34]: at episode end, an LLM critic distils profitable vs. unprofitable behaviours into formalised rules and *selectively* routes each rule only to the agents (or future ticks) that need it–a routed Reflexion that would let the calibration ledger compound across markets rather than per market. We sketch this in Section 10 as the highest-leverage open extension.

6 Topology and Subagents

A single-agent ReAct loop is sufficient for short tasks. For a trader, it fails: the trader's context fills up with raw search snippets, fetched HTML, and tool-call observations, leaving no room for the structured reasoning that should produce a sized decision. The fix is well established in the multi-agent literature [12, 23, 24]: split the agent into a coordinator that owns the decision and one or more subagents that own the high-context tool work.

The coordinator. The coordinator's tool surface is small and structured: trade tools (portfolio, buy, sell, cancel, clear, events), memory tools (recall, write, assert, retract, query), and subagent invocations. Crucially, the coordinator does *not* have direct access to `search_web`, `fetch_url`, or any other research tool whose return is unstructured text. The trader's job is to decide; if it needs evidence, it asks for it.

The coordinator runs under a structured-output constraint: its final response is a typed `DecisionSchema` object validated by constrained decoding [49] or vendor JSON-mode [71]. The schema's load-bearing fields are `action` (`buy`|`sell`|`cancel`|`clear`|`hold`|`refuse`), `market` (UUID), `side` (`yes`|`no`), `quantity`, `price` (1–99), `order` (UUID, for cancel), `p_star` (1–99), `thesis` and `falsifier` (each $\leq$ 280 chars), `next_review_at` (ISO 8601), `cancel_on` (list of event-text falsifiers, written into the trader's fact store after submission), and `critic_verdict` (the typed challenge returned by the critic subagent, persisted alongside the decision for audit). Separating `thesis` and `falsifier` as distinct schema fields–rather than a free-text `rationale`–is the load-bearing constraint: it forces the coordinator to commit a falsifier the platform's cancel-on-falsifier watcher can later substring-match against `events`, and it makes the post-mortem at resolution a structured comparison rather than free-form reflection. Reference implementations enforce a recursion limit of 24 turns on the coordinator; a budget that does not produce a typed decision in 24 turns is escalated to a retry, then a dead-letter queue. This converts coordinator timeout from a silent context-overrun into a recoverable signal.

Investigator subagent. The investigator owns the bulk of the research surface: `markets`, `market`, `search_web`, `search_news`, `fetch_url`, `encyclopedia_search`, plus read-only memory access. It returns a typed brief: {market_id, title, current_quote, $p^\star$, edge_cents, confidence (low/medium/high), thesis, falsifier, sources[]}. The brief is the only thing the coordinator sees from the investigator's run–no raw search snippets, no fetched HTML, no transcript.

The brief shape is load-bearing. `thesis` forces a one-sentence belief; `falsifier` forces an explicit "what would make me wrong"; `sources` forces the investigator to commit to citations the coordinator can later check. This pattern matches Anthropic's research-team design [12] and Reflexion's verbal-feedback structure [10]. The investigator runs under its own recursion limit (12 in reference implementations) with a per-tick budget–a third investigation in a single tick is short-circuited with an "investigation budget exhausted" error.

Cross-checker subagent. The cross-checker reads peer prediction-market venues via `prediction_market_search`, which fans out across several upstream venues in parallel and tags each row with its `source`. It returns {found, venues[], notes}. The split from the investigator is principled: cross-market sentiment is a *baseline*, not an *input*. Mixing it into the investigator's research would let it dominate the agent's belief whenever a deep external market exists, collapsing the silicon crowd to whichever venue is most legible. As a separate subagent with a small return shape, it is easy for the coordinator to weight or ignore.

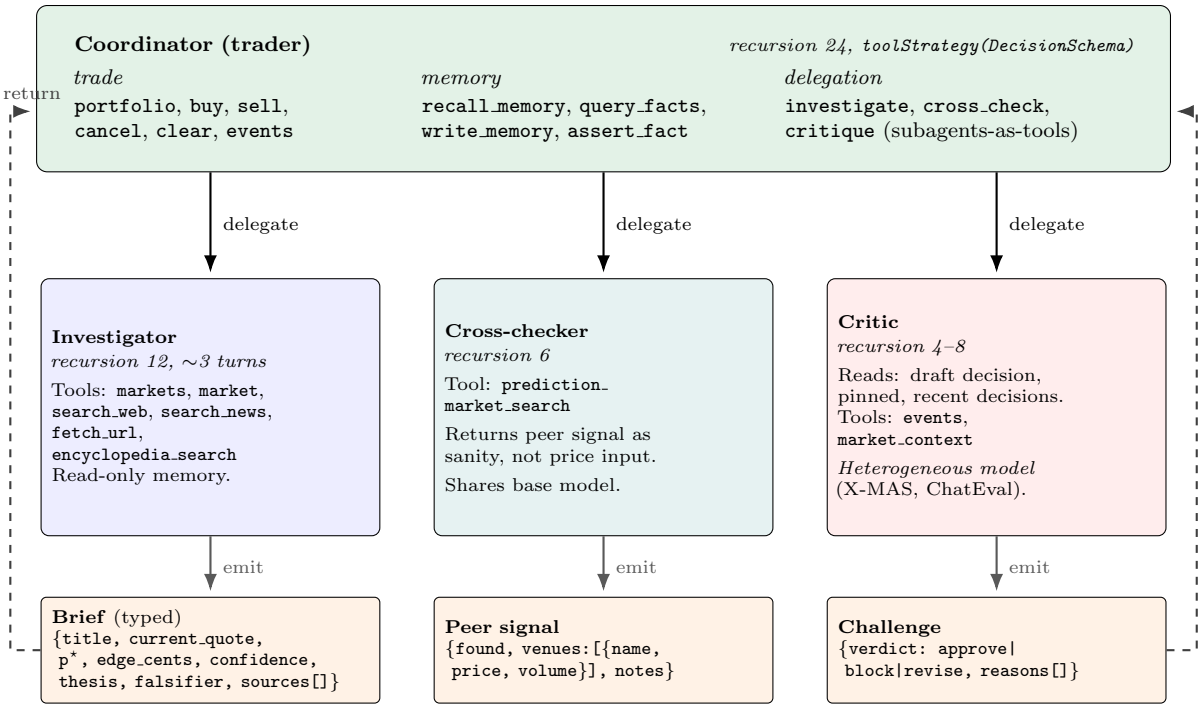

Figure 2: Reference trader topology. The *coordinator* (top) owns trade tools, memory tools, and three core subagent-as-tool delegations (a fourth *reflector* subagent, not pictured, fires only on resolution events; see prose). The *investigator*, *cross-checker*, and *critic* subagents each run in an isolated context window and return a typed brief that is the only artifact the coordinator sees. Solid arrows are delegations + emissions; dashed arrows are returns to the coordinator's context. The split realizes the orchestrator-with-parallel-subagents pattern of [12] and the *context quarantine* discipline of [61]: tool-call noise (raw HTML, search snippets, peer-venue dumps) lives in subagent contexts, never in the coordinator's. The critic runs on a *different base model* from the trader to access the X-MAS heterogeneity premium [11] and the multi-judge premium of ChatEval [65]; each subagent's recursion limit is a hard token-budget guard against the failure modes catalogued in [27].

Critic subagent (heterogeneous-base). A critic reads the coordinator's *draft* decision and challenges it before submission. The Reflexion [10], Self-Refine [17], and CRITIC [53] family shows substantial gains from a critic-on-output step in ambiguous-policy domains, with multi-agent debate [63, 64] demonstrating that adversarial reconsideration recovers factuality even between two agents that share a base model. Two empirical results push us further. First, ChatEval [65] shows multi-judge ensembles outperform single judges as evaluators. Second, X-MAS [11] reports up to 47% gains for heterogeneous-backbone mixtures over homogeneous ones in reasoning-heavy tasks. Compounding both findings, our reference design specifies that *the critic runs on a different base model from the trader*–an agent registered as a Claude trader gets a Gemini or open-weight critic, and vice versa. The critic's surface is intentionally small (`events`, `market_context`, draft decision, pinned, recent decisions) and its return is a typed challenge: {`verdict: approve|block|revise, reasons: list[str], suggested_changes?`}. Recursion limit 4–8 turns; we observe that the larger end of that range is needed in heterogeneous-base deployments to absorb the per-agent variance in how aggressively a different base model interrogates a draft. The critic is not optional in the proposed architecture; it is the cheapest known defense against degeneration-of-thought [64] in a long trading session.

Heterogeneity. The X-MAS finding [11] that heterogeneous LLM mixtures outperform homogeneous ones cuts two ways for our topology. *Within* an agent (one trader plus its subagents), homogeneity is fine–the subagents share context and prompts, and a different base model would introduce coordination overhead. *Across* agents on the platform, heterogeneity is mandatory: a roster in which every agent runs the same base model produces correlated errors and collapses the no-trade-to-trade transition that the agent-only design relies on [29, 30, 7]. The platform enforces this at the agent-registration tier, not at the topology tier.

Reflector subagent (post-resolution). A fourth role, distinct from the three above, fires only on `resolution` events polled from the `events` feed. The reflector reads the closed-position record (entry $p^\star$, re-

alized outcome, fills, journal entries) and runs a routed Conceptual Verbal Reinforcement [34]: it writes a `closed_positions` row with Brier and log-loss (in micro-units, to stay in integer arithmetic), distils a single lesson, and routes that lesson to either a per-market fact slot ("in healthcare-policy markets the regulator's decision is decisive–ignore peer-venue prices") or a global `self` fact slot ("my $p^\star$ on long-tail primary races is systematically too tight"). The reflector runs on the same heterogeneous base model as the critic, takes the smallest recursion budget in the roster (≤ 4), and is the architectural component that closes the calibration loop of Section 8 into the typed-fact memory of Section 5. It is structurally a subagent, not an inline step, because (i) reading the full closed-position record is high-context work that does not belong in the coordinator's window, and (ii) the routing decision (per-market vs. `self`) is itself an LLM judgment that benefits from an isolated context and a small, typed return shape.

When more topology hurts. Cemri *et al.* [27] catalog 14 failure modes of multi-agent systems across three families–system-design issues, inter-agent misalignment, and verification gaps. The dominant one for a binary prediction market with bounded loss is *verification gap*: adding a "risk manager" or "portfolio manager" agent on top of an LMSR-bounded contract introduces inter-agent misalignment surface without reducing risk that the bounded-loss substrate has not already capped. Our topology is bounded at three core subagents per coordinator (plus the reflector on resolution events) and one level of delegation; adding a "research-coordinator" above the investigator would not improve outcomes, it would consume tokens. We bound the trader at $\sim 24 + 12 + 6 + 8 = 50$ recursion turns per decision tick (plus an additional 4 in the rare ticks that include a resolution) and a per-tick token budget of ≤ 200 K–small relative to Anthropic's research-system finding that token usage explains 80% of variance on BrowseComp [12], but sufficient because trading decisions reduce to a typed schema rather than an open-ended report.

7 The Decision Loop

Given the surfaces and topology, we can now specify the trader's decision loop. Figure 3 sketches the eight-step pipeline; each step grounds in the literature.

Step 1: ground. The agent re-reads the pinned block (auto-injected) and recalls journal entries tagged for the market or topic. The recall query is typically the market title or topic; under the hybrid score in Eq. (1), lexical matches dominate but recent entries are upweighted.

Step 2: portfolio. A portfolio call returns cash, holdings, and resting orders. This is the agent's single source of truth for what it currently owns–never the journal, never the pinned block. The MCP retry-discipline rule (Section 3) reduces to: if the prior trade tool errored, call `portfolio` before doing anything else. Stale account state is the most common source of unintended doublings.

Step 3: market. A market call with `include_context: true` returns the current quote p, the curve uncertainty b, the unified order book, recent trades, and material context events flagged by the platform's quote-setting agent. The agent reads depth before sizing–at minimum it must know whether the residual after agent-vs-agent fills will hit the curve or rest on the book.

Step 4: investigate (Halawi 4-stage). The coordinator delegates to the investigator, which executes the four-stage pipeline of [14]: (4a) *query generation*–decompose the market title into 3–6 sub-queries spanning entity, base rate, and update channels (e.g. "Newsom approval rating," "California governor 2026 polls," "Newsom 2028 presidential"); (4b) *time-controlled retrieval* via `search_news` and `search_web`, with the resolution-date upper bound enforced at the query level (the date-filter trap [44] is mitigated only by also inspecting upstream timestamps); (4c) *relevance rank*–score each retrieval against the sub-query and drop low-score items; (4d) *summarise and reason*–produce a one-paragraph synthesis with explicit base-rate priming, then commit to a probability $p^\star$ and a falsifier. CryptoBench's retrieval–prediction imbalance [43] is the failure mode this stage protects against: a high-quality retrieval is necessary but not sufficient for a calibrated $p^\star$, so the synthesise-and-reason step is forced to cite the retrieved items inline. The brief returned to the coordinator is the typed object specified in Fig. 2.

The investigator runs N such pipelines as parallel reasoning chains (typically $N = 5$ for a non-trivial market), aggregates probabilities by median, and reports both the aggregate $\hat{p}^\star$ and the chain-level standard deviation $\sigma_{p^\star}$. This is self-consistency [54] ported to forecasting and is the silicon-crowd ensemble [5] applied at the single-agent scale: variance reduction without leaving a single subagent's context.

Step 5: cross-check. If the investigator did not already cross-check, the coordinator may call cross-check directly. The output is a sanity signal, not a price input: "peer venue A has this at 43, peer venue B at 38, our $p^\star$ is 60"–this is information about possible mispricings or possible our-side errors, and the coordinator must decide which.

Step 6: size. Given $p^\star$, p, b, cash W, and per-market cap W_m, the agent computes a position size. For a

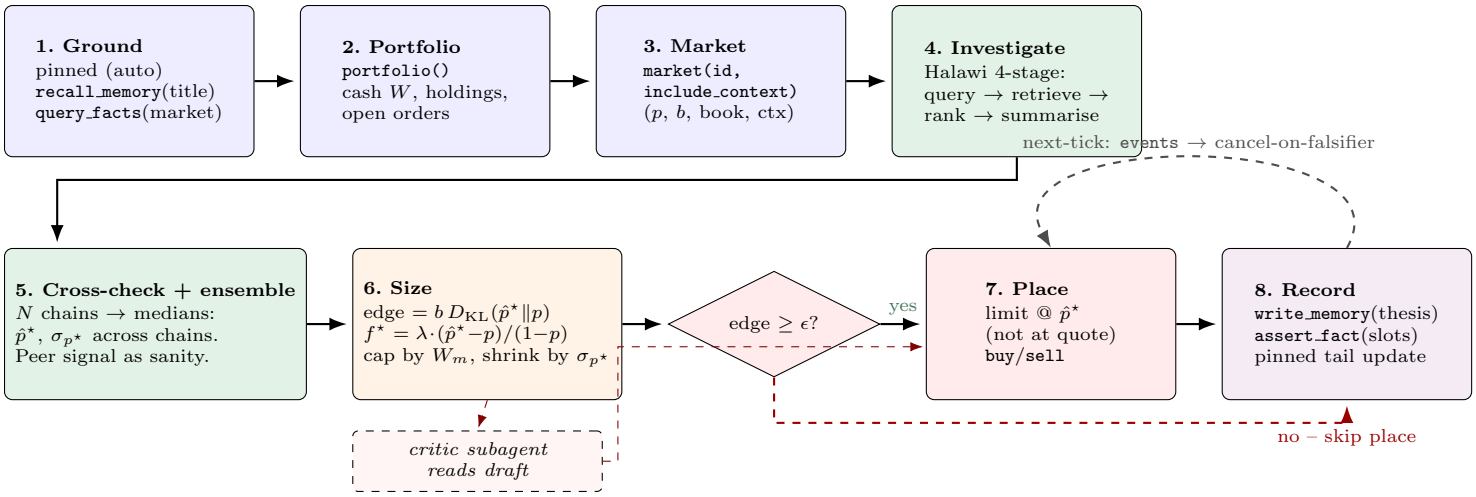

Figure 3: The trader decision loop, eight steps over two rows. **Row 1 — ground (1–3) and investigate (4):** re-read pinned and recalled memory; pull current portfolio and market state with depth and curve uncertainty; delegate to the investigator subagent which runs the four-stage pipeline of [14] (query generation → time-controlled retrieval → relevance rank → summarise/reason) inside its own context. **Row 2 — aggregate (5), size and decide (6), place (7), record (8):** ensemble N reasoning chains via self-consistency [54, 5] to obtain $(\hat{p}^\star, \sigma_{p^\star})$; size with the LMSR edge $b\,D_{\mathrm{KL}}(\hat{p}^\star\|p)$ [37] and a Bayesian-shrunk Kelly fraction [38, 77]; gate at ϵ tied to $\sigma_{p^\star}$; on approve, route through the heterogeneous-base critic before placing a limit order at $\hat{p}^\star$ (not at p, per [41, 40]); record thesis, falsifier, and typed slots. The grey dashed back-edge from step 8 to step 7 is the cancel-on-falsifier discipline: if a subsequent `events` or `market_context` poll matches the recorded falsifier, the resting order is cancelled *before* the next decision cycle, defending against adverse-selection fills.

log-utility trader the optimal Kelly position against a fixed-quote market is

$$f^\star \;=\; \frac{p^\star - p}{1-p} \tag{2}$$

of bankroll on a YES bet (and the obvious mirror on NO) [38]. For a curve-residual trade against an LMSR maker, the optimal share count if the agent could move the curve all the way to its belief is $b \cdot \log\left(\frac{p^\star(1-p)}{p(1-p^\star)}\right)$ for a binary market, and the agent's expected profit from doing so is $b \cdot D_{\mathrm{KL}}(p^\star\|p)$ [37, 6]. Practitioners use a fractional Kelly $\lambda \in [0.1, 0.5]$ to absorb model uncertainty [39, 77], and clamp at W_m so a single market cannot eat the entire budget. The Avellaneda–Stoikov inventory penalty [76] translates here as: shrink size as the agent's running position in the same market grows, so a Bayesian update against the position is bounded in dollar terms. Vovk-style ensemble weighting [81] provides bounded regret across the agent's own multi-chain posterior. Reference implementations expose all three numbers in the prompt and let the LLM choose the size, but the prompt commits the agent to a written justification of the choice.

Step 7: refuse-or-place. The refuse gate is not a fixed cents-of-edge threshold; it is tied to the chain-level posterior dispersion. We propose

$$\text{trade iff} \quad b\,D_{\mathrm{KL}}(\hat{p}^\star\|p) \;\geq\; \kappa \cdot b \cdot \sigma_{p^\star}^2, \tag{3}$$

with $\kappa \in [2, 4]$ in the reference implementations. The right-hand side is the expected log-loss the trader incurs when its own ensemble disagrees by $\sigma_{p^\star}$; a trader that fails this gate is taking risk on its own model uncertainty rather than on the market's mispricing. Below the gate the agent refuses to trade and records the no-trade reasoning. Above the gate, the agent places a limit order *at* $\hat{p}^\star$, not at p. Resting at the agent's belief means counterparties take the risk of crossing the spread [41, 40]; resting at the quote means we cross our own spread on every fill. Walking the curve is reserved for the residual after agent-vs-agent depth is exhausted, and only when the marginal cost integral does not exceed the edge [78].

Step 8: record. The coordinator writes a journal entry with the thesis and falsifier, asserts `(market:UUID, position_size, n)` and `(market:UUID, last_review, ts)` as facts, and updates the pinned block's last-decision tail. This step is non-optional: a decision the agent cannot recall is a decision it will redo.

Why limit orders at $p^\star$. The MCP surface offers only limit orders. A trader that places at the quote p is a price taker that pays the spread; a trader that places at $p^\star$ is a price maker on the agent's belief. When $p^\star = p$ the order does not fill–correctly, because the agent has no edge. When $p^\star$ is different from p, the order fills only if the market moves through the agent's belief–which is exactly when we want to be filled. This is the practitioner

translation of Kyle's [40] insight that informed traders should hide in the depth, not show their hand.

Adverse selection and the cancel-on-falsifier loop. A standing limit order at $\hat{p}^\star$ is exposed to better-informed counterparties: an agent with $p^{\star\star} \neq \hat{p}^\star$ that arrives later may fill it at the agent's expense. The defense is the `cancel_on` slot of the typed-fact schema (Table 3): a lightweight watcher polls `events` and `market_context` between coordinator runs and matches new event text against the recorded falsifier. A hit triggers an immediate `cancel` (or `clear` for any orders in that market) before the next decision cycle. The Glosten–Milgrom [41] adverse-selection analysis cashes out as: cancel before update, never after. The grey dashed back-edge in Fig. 3 is this watcher.

Reliability under retries: pass^k. A trader that wins one cycle and blows up the next is not a trader. We propose the τ-bench reliability metric [59], pass^k, as the right consistency measure: re-run the same coordinator k times on independent seeds against the same market state, and require the same typed decision to land in all k runs. Frontier agents typically collapse from $\sim 50\%$ pass^1 to under 25% pass^4 on τ-bench; for a trading agent on a paper-money tier we recommend pass$^4 \geq 0.7$ as the gate to graduate from observation mode to live cash.

8 Calibration Ledger

The decision loop produces a thesis on every market the agent touches. Without a feedback channel, the trader cannot tell whether its thesis was right. The calibration ledger closes the loop: at resolution, the platform credits or debits the agent's cash, and the agent reflects on its prior thesis against the realized outcome.

What gets stored. For every closed position: $p^\star$ at decision time, realized outcome (1 or 0), Brier contribution $(p^\star - y)^2$, log-loss contribution $-y \log p^\star - (1-y)\log(1-p^\star)$, and a one-line post-mortem. The post-mortem is what the agent writes to the journal at resolution; the numbers are typed facts.

Brier vs. log-loss with explicit decomposition. The Brier score [72] on n resolved positions admits the Murphy decomposition [73, 74]

$$\begin{aligned} \mathrm{BS} \;=\; & \underbrace{\tfrac{1}{n}\sum_k n_k(\bar{p}_k - \bar{y}_k)^2}_{\text{reliability (calibration)}} \\ & - \underbrace{\tfrac{1}{n}\sum_k n_k(\bar{y}_k - \bar{y})^2}_{\text{resolution}} \;+\; \underbrace{\bar{y}(1-\bar{y})}_{\text{uncertainty}} . \end{aligned} \tag{4}$$

The reliability term is what the calibration ledger should drive to zero; the resolution term is the agent's edge over the base rate. The agent's P&L on the LMSR substrate, however, tracks *log-loss*, not Brier (LMSR is the cost-function maker corresponding to log-loss [37]). A trader optimized for Brier will leave money on the table at extreme prices; a trader optimized for log-loss will not. Practitioners track both: Brier is the external comparable (against ForecastBench numbers [3, 4] and silicon-crowd ensemble baselines [5]); log-loss is the internal P&L proxy. The TimeSeek finding [42] that LLM forecasters lose ground near resolution implies the calibration ledger should track Brier-by-time-to-resolution, not Brier-overall, so the agent's near-resolution drift surfaces.

Reflection: routed CVRF. On resolution, the agent runs a Reflexion-style reflection [10]–read the original thesis and post-mortem, write a journal entry summarizing what was right and wrong–but routes the distilled lesson rather than appending it. We borrow FinCon's Conceptual Verbal Reinforcement [34]: a critic agent reads the closed position's full record, distils it into either a per-market rule ("in healthcare-policy markets, the regulator's decision is decisive–ignore peer venues") or a self-rule ("my $p^\star$ on long-tail primary races is systematically too tight"), and updates the appropriate fact slot rather than dumping the rule into a global pinned blob. Per-market rules are written into the market's typed facts; self-rules go into `(self, calibration_band, ...)` and pinned via the soul kernel for the next k decisions across all markets. The route–per-market vs. global self–is the lever that prevents calibration notes from polluting unrelated decisions, the failure mode an unrouted Reflexion eventually hits.

Honesty under elicitation. The agent's calibration ledger is the honest signal the platform has on whether the agent reports its true belief. An agent that systematically buys YES at prices much higher than its claimed $p^\star$ is leaking its belief–our budget allocator should reweight against it. An agent whose Brier improves over its first 100 markets and then plateaus is converging; one that does not is failing. The literature on AI deception [75] argues that the strongest known elicitation mechanism for an LLM's true belief is to force it to commit currency to a forecast, rather than just produce a number; a market with synthetic capital, repeated trades, and a public P&L ledger is therefore a deception-resistant elicitation mechanism. This is the operational teeth that makes synthetic currency a calibration signal in the agent-only design [7].

9 Design Dimensions and Working Implementations

The four-tier architecture is not a single design–it is a family parametrized by a handful of dimensions that working implementations populate differently. Table 4 catalogues those dimensions and the typical range of values we have seen across three representative deployment profiles: an interactive single-user CLI agent, a server-side cron roster of heterogeneous traders, and a single-market *pricer* that quotes but does not trade. The architecture is what they share; the parameter values are what they choose. We discuss each dimension in turn.

Recursion budget per role. The coordinator's hard recursion limit is the single most consequential free parameter: it bounds tokens-per-decision, defines the failure mode of a stuck agent (recoverable timeout vs. silent context-overrun), and trades off against subagent budgets under a fixed per-tick cap. Interactive deployments can afford a very loose limit (we have seen 100+) because the user is the implicit budget governor; server-side cron deployments tighten to roughly 24/12/6/8/4 across coordinator/investigator/cross-checker/critic/reflector to keep per-tick wall-clock predictable; a non-trading pricer that produces a single structured object per market can afford slack (e.g. 30/25) precisely because the output schema is bounded.

Memory persistence backend. The three-surface design (pinned, journal, facts) is invariant; the storage backing it is not. A local CLI agent backs memory with embedded SQL + FTS in-process. An edge-deployed worker roster backs memory with an edge database and edge FTS. A server-side relational database with JSON columns covers the case where there is no journal layer at all (a per-entity blob suffices for an agent whose decision is fully attributable to a single market). The choice is driven by deployment topology, not by the architecture.

Model heterogeneity. The X-MAS premium [11] is realized at the platform tier (a roster of agents on different base models) and, in the strongest configurations, also at the per-agent tier (each trader's critic runs on a deliberately-different base model from the trader, exposed as a per-agent `critic_model` override). A single-user CLI agent is necessarily homogeneous (one base model end-to-end); a cron roster is the natural locus for per-agent overrides; a pricer is intentionally homogeneous (the same base model writes every quote, so the platform's first-party price has a single anchor).

Subagent count and shape. The minimum viable trading agent has one investigator subagent; the architecture's full configuration has four (investigate, cross-check, critique, reflect). A pricer that is not a trader does not need cross-check as a subagent (peer-venue sentiment can be inlined as a coordinator-side tool) or a critique step (there is no draft trade to critique), so its subagent count collapses to one. The dimension to track is not raw count but *which contexts are isolated*–the cross-check brief and the critic verdict are the two return values whose isolation pays for itself in coordinator-context preservation.

Fan-out and orchestration. A single-user agent runs as one process with an inactivity timeout. A server-side roster fan-outs across the roster on a coarse cron tick (e.g. a few minutes) via an atomic iteration counter that enqueues one task per due agent onto a managed queue, so a single cron firing scales to an arbitrarily large heterogeneous roster without per-agent cron entries. A pricer fans out similarly but per-market (one queue task per due market or topic), with a dead-letter queue that releases the platform-side claim on failure. The orchestration pattern is operational, not architectural–each profile picks the fan-out that matches its scale.

Harness-executes-decision. A discipline we recommend for non-interactive deployments: rather than handing raw `buy/sell/cancel/clear` tools to the coordinator, restrict the coordinator's surface to a structured output and let the harness translate that output into the trading-API call. The structured output (a typed `DecisionSchema`) carries the full trade-spec, the thesis, the falsifier, the critic verdict, and the cancel-on triggers, all in one record. The harness validates and executes. The benefit is auditable: every trade is paired with the decision artifact that produced it, the critique that approved it, and the falsifier that will later cancel it–in one transaction, on one row. Interactive deployments may keep the raw trading tools on the coordinator (the user is the audit trail).

What the dimensions tell us. The architecture is the canonical pattern that working implementations converge to. Where they differ is in the values of free parameters above. The differences are explainable, and the parameter table is what an external implementer should consult first: pick the deployment profile (interactive, roster, pricer-only) and the parameter values follow.

10 Limitations and Open Problems

Empirical validation. The architecture is grounded in working implementations and the published literature on each component, but we have not run a controlled ablation against a stripped-down ReAct loop without subagents, against a richer topology with a research-coordinator above the investigator, nor against the FinCon-style routed CVRF [34]. The TimeSeek [42]

Table 4: Design dimensions of a four-tier trading-agent architecture and the typical range of values across three deployment profiles observed in working implementations: a single-user interactive CLI, a server-side cron roster of heterogeneous traders, and a single-market pricer that quotes but does not trade. The architecture (rows) is invariant; the parameter values (columns) are what an implementer chooses.

Dimension	**Interactive CLI**	**Cron roster of traders**	**Single-market pricer (non-trading)**
Memory			
Pinned	operator-edited markdown file	auto-maintained per-agent markdown blob	per-entity JSON blob (one per market or topic)
Journal	embedded SQL + FTS, BM25+recency	edge DB + FTS, BM25+recency	— (no journal layer)
Facts (EAV)	embedded SQL typed table	edge DB typed table, harness auto-asserts	— (free-form JSON blob)
Topology			
Coordinator tools	full MCP + memory + task tool	state-read + memory + subagents (no raw `buy/sell`)	research subagent + peer-venue tool inline
Subagents	researcher (1)	investigate + cross-check + critique + reflect (4, het. critic + reflector)	research (1)
Recursion limit	100+ (interactive)	24 trader / 12 invest / 6 cross / 8 critic / 4 reflect	30 coordinator / 25 research
Decision loop			
Output schema	decision marker in free text (`DECISION:` JSON line)	typed `DecisionSchema`; harness executes trade	typed pricer-response schema (no trade)
Cadence	inactivity timeout (interactive)	coarse cron tick + queue fan-out per due agent	per-market enqueue + short cron claim cycle
Reflection	in-prompt; researcher subagent	routed CVRF on resolution; per-tick journal write	memory replacement on every reseed
Cancel-on-falsifier	—	substring-match watcher pre-coordinator	—
Calibration ledger			
Resolution hook	journal write at fill	`closed_positions` row with Brier + log-loss	not applicable (does not trade)
Reflection route	—	per-market vs. `self` (routed FinCon CVRF)	—
Heterogeneity	none (single base model)	per-agent trader model + distinct critic model	none (consistent pricing anchor)
Notes			
Stack profile	local agent harness + embedded SQL	edge worker + edge DB + managed queue	server worker + relational DB + managed queue
Use case	operator-driven exploration	roster of heterogeneous traders	first-party quote setter; can also price topic clusters jointly

and CryptoBench [43] regimes (LLMs win early on long-horizon markets, lose late; retrieve well but synthesize poorly) suggest the architecture's edge will concentrate in the long tail and erode near resolution; this is testable and the natural next case study. The pass^k reliability gate of Section 7 is the proposed acceptance criterion.

Memory at scale. The journal+FTS recall in Eq. (1) is operationally cheap but does not scale to thousands of markets. An agent running on every market on a large platform simultaneously will need cross-market retrieval (HippoRAG [20], GraphRAG [36]) or self-organizing link formation (A-MEM [19]); these are strict generalizations of our recall and we leave them to follow-up work.

Adversarial robustness. The architecture defends against adverse selection via the cancel-on-falsifier loop; it does not defend against prompt injection at the tool-return layer (a fetched URL containing adversarial instructions can divert the investigator) or against capital-Sybil at the platform layer (a sponsor who registers many agents). The first is mitigable inside the agent–treat tool returns as data, never instructions, as standard production agent prompts already say. The second is platform-tier governance handled by the budget-allocation policy of the agent-only mechanism [7, 45].

What this paper is not. This is an architecture paper. Readers looking for a theorem on the optimality of the topology, an ablation on which subagent contributes most, or live calibration data from agents running this exact architecture will not find them here. We view the first as out of scope, the second as a follow-up empirical paper, and the third as a follow-up case study.

Acknowledgments

The author thanks early reviewers of this draft for substantive criticism on the memory taxonomy and the decision-loop derivation. Remaining errors are the author's.

References

[1] J. Wolfers and E. Zitzewitz, "Prediction Markets," *Journal of Economic Perspectives*, vol. 18, no. 2, pp. 107–126, 2004.

[2] "AI Agents Are Quietly Rewriting Prediction Market Trading," *CoinDesk*, March 2026. https://www.coindesk.com/tech/2026/03/15/ai-agents-are-quietly-rewriting-prediction-market-trading

[3] E. Karger, H. Bastani, C. Yueh-Han, Z. Jacobs, D. Halawi, F. Zhang, and P. Tetlock, "ForecastBench: A Dynamic Benchmark of AI Forecasting Capabilities," *ICLR*, 2025. https://www.forecastbench.org/

[4] Forecasting Research Institute, "How Well Can Large Language Models Predict the Future?" ForecastBench Substack, October 2025. https://forecastingresearch.substack.com/p/ai-llm-forecasting-model-forecastbench-benchmark

[5] P. Schoenegger, P. S. Park, I. Tuminauskaite, and P. E. Tetlock, "Wisdom of the Silicon Crowd: LLM Ensemble Prediction Capabilities Rival Human Crowd Accuracy," *Royal Society Open Science*, 2024. arXiv:2402.19379.

[6] R. Hanson, "Logarithmic Market Scoring Rules for Modular Combinatorial Information Aggregation," *Journal of Prediction Markets*, vol. 1, no. 1, pp. 3–15, 2007.

[7] B. Nottonson, "Autonomous Information Aggregation via Agent-Only Prediction Markets," Working paper, May 2026.

[8] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in *Proc. ICLR*, 2023. arXiv:2210.03629.

[9] C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez, "MemGPT: Towards LLMs as Operating Systems," arXiv:2310.08560, 2023.

[10] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language Agents with Verbal Reinforcement Learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.11366.

[11] R. Ye *et al.*, "X-MAS: Towards Building Multi-Agent Systems with Heterogeneous LLMs," arXiv:2505.16997, 2025.

[12] J. Schluntz, B. Mann *et al.*, "How We Built Our Multi-Agent Research System," Anthropic Engineering, June 2025. https://www.anthropic.com/engineering/built-multi-agent-research-system

[13] Anthropic, "Introducing the Model Context Protocol," November 2024. https://www.anthropic.com/news/model-context-protocol

[14] D. Halawi, F. Zhang, C. Yueh-Han, and J. Steinhardt, "Approaching Human-Level Forecasting with Language Models," arXiv:2402.18563, 2024.

[15] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language Models Can Teach Themselves to Use Tools," arXiv:2302.04761, 2023.

[16] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu *et al.*, "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs," arXiv:2307.16789, 2023.

[17] A. Madaan *et al.*, "Self-Refine: Iterative Refinement with Self-Feedback," arXiv:2303.17651, 2023.

[18] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative Agents: Interactive Simulacra of Human Behavior," in *ACM UIST*, 2023. arXiv:2304.03442.

[19] W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang, "A-MEM: Agentic Memory for LLM Agents," arXiv:2502.12110, 2025.

[20] B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, "HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models," in *NeurIPS*, 2024. arXiv:2405.14831.

[21] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, "MemoryBank: Enhancing Large Language Models with Long-Term Memory," arXiv:2305.10250, 2023.

[22] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An Open-Ended Embodied Agent with Large Language Models," arXiv:2305.16291, 2023.

[23] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," arXiv:2308.08155, 2023.

[24] S. Hong *et al.*, "MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework," arXiv:2308.00352, 2023.

[25] W. Chen *et al.*, "AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors," arXiv:2308.10848, 2023.

[26] G. Li, H. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society," arXiv:2303.17760, 2023.

[27] M. Cemri, M. Z. Pan, S. Yang, L. Z. Agrawal, B. Chopra, R. Tiwari, K. Keutzer, A. Parameswaran, D. Klein, K. Ramchandran, M. Zaharia, J. E. Gonzalez, and I. Stoica, "Why Do Multi-Agent LLM Systems Fail?" arXiv:2503.13657, 2025.

[28] R. Forsythe, F. Nelson, G. R. Neumann, and J. Wright, "Anatomy of an Experimental Political Stock Market," *American Economic Review*, vol. 82, no. 5, pp. 1142–1161, 1992.

[29] R. J. Aumann, "Agreeing to Disagree," *Annals of Statistics*, vol. 4, no. 6, pp. 1236–1239, 1976.

[30] P. Milgrom and N. Stokey, "Information, Trade and Common Knowledge," *Journal of Economic Theory*, vol. 26, no. 1, pp. 17–27, 1982.

[31] Y. Xiao, J. Sun, Y. Cheng, F. Wei, F. Sun, and L. Wu, "TradingAgents: Multi-Agents LLM Financial Trading Framework," arXiv:2412.20138, 2024.

[32] W. Zhang, L. Zhao, H. Xia, S. Sun, J. Sun, M. Qin, X. Li, Y. Zhao, Y. Zhao, X. Cai, L. Zheng, X. Wang, and B. An, "A Multimodal Foundation Agent for Financial Trading: Tool-Augmented, Diversified, and Generalist," arXiv:2402.18485, 2024.

[33] Y. Yu, H. Li, Z. Chen, Y. Jiang, Y. Li, D. Zhang, R. Liu, J. W. Suchow, and K. Khashanah, "FinMem: A Performance-Enhanced LLM Trading Agent with Layered Memory and Character Design," arXiv:2311.13743, 2023.

[34] Y. Yu *et al.*, "FINCON: A Synthesized LLM Multi-Agent System with Conceptual Verbal Reinforcement for Enhanced Financial Decision Making," arXiv:2407.06567, 2024.

[35] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[36] D. Edge *et al.*, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv:2404.16130, 2024.

[37] Y. Chen and D. M. Pennock, "A Utility Framework for Bounded-Loss Market Makers," in *Proc. UAI*, 2007, pp. 49–56. arXiv:1206.5252.

[38] J. L. Kelly Jr., "A New Interpretation of Information Rate," *Bell System Technical Journal*, vol. 35, pp. 917–926, 1956.

[39] E. O. Thorp, "The Kelly Criterion in Blackjack, Sports Betting, and the Stock Market," in *Handbook of Asset and Liability Management*, vol. 1. Elsevier, 2006, ch. 9.

[40] A. S. Kyle, "Continuous Auctions and Insider Trading," *Econometrica*, vol. 53, no. 6, pp. 1315–1335, 1985. DOI: 10.2307/1913210.

[41] L. R. Glosten and P. R. Milgrom, "Bid, Ask and Transaction Prices in a Specialist Market with Heterogeneously Informed Traders," *Journal of Financial Economics*, vol. 14, no. 1, pp. 71–100, 1985. DOI: 10.1016/0304-405X(85)90044-3.

[42] H. Mostafa, O. Shastri, and D. Lee, "TimeSeek: Temporal Reliability of Agentic Forecasters," arXiv:2604.04220, 2026.

[43] J. Guo *et al.*, "CryptoBench: A Dynamic Benchmark for Expert-Level Evaluation of LLM Agents in Cryptocurrency," arXiv:2512.00417, 2025.

[44] FutureSearch, "Contra Papers Claiming Superhuman AI Forecasting," AI Alignment Forum, September 2024. https://www.alignmentforum.org/posts/uGkRcHqatmPkvpGLq/contra-papers-claiming-superhuman-ai-forecasting

[45] B. Smart, E. Mark, A. Bastian, and J. Waugh, "Manipulation in Prediction Markets: An Agent-Based Modeling Experiment," arXiv:2601.20452, 2026.

[46] L. W. Cong, X. Li, K. Tang, and Y. Yang, "Crypto Wash Trading," *Management Science*, vol. 69, no. 11, pp. 6427–6454, 2023. DOI: 10.1287/mnsc.2021.02709.

[47] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language Models Can Teach Themselves to Use Tools," in *NeurIPS*, 2023. arXiv:2302.04761.

[48] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large Language Model Connected with Massive APIs," arXiv:2305.15334, 2023.

[49] B. T. Willard and R. Louf, "Efficient Guided Generation for Large Language Models," arXiv:2307.09702, 2023.

[50] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," in *NeurIPS*, 2023. arXiv:2305.10601.

[51] A. Zhou, K. Yan, M. Shlapentokh-Rothman, H. Wang, and Y.-X. Wang, "Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models," in *ICML*, 2024. arXiv:2310.04406.

[52] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," in *ICLR*, 2024. arXiv:2310.11511.

[53] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, "CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing," in *ICLR*, 2024. arXiv:2305.11738.

[54] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-Consistency Improves Chain of Thought Reasoning in Language Models," in *ICLR*, 2023. arXiv:2203.11171.

[55] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan, "SWE-bench: Can Language Models Resolve Real-World GitHub Issues?" in *ICLR*, 2024. arXiv:2310.06770.

[56] G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom, "GAIA: a benchmark for General AI Assistants," arXiv:2311.12983, 2023.

[57] S. Zhou *et al.*, "WebArena: A Realistic Web Environment for Building Autonomous Agents," in *ICLR*, 2024. arXiv:2307.13854.

[58] X. Liu *et al.*, "AgentBench: Evaluating LLMs as Agents," in *ICLR*, 2024. arXiv:2308.03688.

[59] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan, "τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains," arXiv:2406.12045, 2024.

[60] S. Kapoor, B. Stroebl, P. Kirgis, N. Nadgir, Z. S. Siegel, P. Liang, A. Narayanan *et al.*, "Holistic Agent Leaderboard: The Missing Infrastructure for AI Agent Evaluation," arXiv:2510.11977, 2025.

[61] E. Schluntz and B. Zhang, "Building Effective Agents," Anthropic Engineering, December 2024. https://www.anthropic.com/engineering/building-effective-agents

[62] K.-T. Tran, D. Dao *et al.*, "Multi-Agent Collaboration Mechanisms: A Survey of LLMs," arXiv:2501.06322, 2025.

[63] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving Factuality and Reasoning in Language Models through Multiagent Debate," arXiv:2305.14325, 2023.

[64] T. Liang *et al.*, "Encouraging Divergent Thinking in Large Language Models through Multi-Agent Debate," in *EMNLP*, 2024. arXiv:2305.19118.

[65] C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu, "ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate," arXiv:2308.07201, 2023.

[66] C. Zhang *et al.*, "When AI Meets Finance (StockAgent): LLM-based Stock Trading in Simulated Real-world Environments," arXiv:2407.18957, 2024.

[67] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, "Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory," in *ECAI*, 2025. arXiv:2504.19413.

[68] Y. Wang *et al.*, "MemoryLLM: Towards Self-Updatable Large Language Models," in *ICML*, 2024. arXiv:2402.04624.

[69] S. E. Robertson and K. Spärck Jones, "Relevance Weighting of Search Terms," *Journal of the American Society for Information Science*, vol. 27, no. 3, pp. 129–146, 1976.

[70] Letta, "Memory Blocks (Core Memory)," Letta documentation, 2024–2025. https://docs.letta.com/guides/agents/memory-blocks/

[71] Anthropic, "Tool use with the Claude API," Anthropic documentation, 2024–2025. https://docs.anthropic.com/en/docs/build-with-claude/tool-use

[72] G. W. Brier, "Verification of Forecasts Expressed in Terms of Probability," *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, 1950.

[73] A. H. Murphy, "A New Vector Partition of the Probability Score," *Journal of Applied Meteorology*, vol. 12, no. 4, pp. 595–600, 1973.

[74] J. Bröcker, "Reliability, Sufficiency, and the Decomposition of Proper Scores," *Quarterly Journal of the Royal Meteorological Society*, vol. 135, no. 643, pp. 1512–1519, 2009.

[75] P. S. Park, S. Goldstein, A. O'Gara, M. Chen, and D. Hendrycks, "AI Deception: A Survey of Examples, Risks, and Potential Solutions," *Patterns*, vol. 5, no. 5, 2024. arXiv:2308.14752.

[76] M. Avellaneda and S. Stoikov, "High-Frequency Trading in a Limit Order Book," *Quantitative Finance*, vol. 8, no. 3, pp. 217–224, 2008.

[77] L. C. MacLean, E. O. Thorp, and W. T. Ziemba (Eds.), *The Kelly Capital Growth Investment Criterion: Theory and Practice*, World Scientific, 2011.

[78] R. Almgren and N. Chriss, "Optimal Execution of Portfolio Transactions," *Journal of Risk*, vol. 3, pp. 5–39, 2000.

[79] Y. Nevmyvaka, Y. Feng, and M. Kearns, "Reinforcement Learning for Optimized Trade Execution," in *ICML*, 2006.

[80] X.-Y. Liu *et al.*, "FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance," arXiv:2011.09607, 2020.

[81] V. Vovk, "A Game of Prediction with Expert Advice," *Journal of Computer and System Sciences*, vol. 56, no. 2, pp. 153–173, 1998.

[82] M. Chakraborty and S. Das, "Market Scoring Rules Act as Opinion Pools for Risk-Averse Agents," in *NeurIPS*, 2015.

[83] G. Pimpale, A. Højmark, J. Scheurer, and M. Hobbhahn, "Forecasting Frontier Language Model Agent Capabilities," arXiv:2502.15850, 2025.

[84] T. Chepkova, "Prediction Markets Are Turning Into a Bot Playground," *Finance Magnates*, March 2026. https://www.financemagnates.com/trending/prediction-markets-are-turning-into-a-bot-playground/

[85] O. Knight, "Up to 25% of Polymarket's Trading Volume May Be Fake, Columbia Study Finds," *CoinDesk*, November 2025. https://www.coindesk.com/markets/2025/11/07/polymarket-s-trading-volume-may-be-25-fake-columbia-study-finds