MAESTRO: What Orchestration Actually Buys — A Taxonomy of Latency, Capability and Supply Mechanisms for Multi-Model Agent Harnesses

O. Serra — Independent Researcher

September 2026

Abstract

Orchestration is sold to harness builders as a way to make agents "smarter," but the mechanisms gathered under that word move two different quantities with two different cost structures, and conflating them is the field's most expensive error. We separate them. **Latency mechanisms** buy wall-clock time at roughly constant token cost by exploiting independence in the task graph; **capability mechanisms** buy accuracy by spending strictly more tokens, and most of them make latency worse. We give a taxonomy of twelve mechanisms across both axes, each stated with the resource it consumes and the structural precondition that must hold for it to pay at all. We then argue the field's central empirical result is negative: mixing models does not reliably beat repeating the best one (Li, J. et al., 2024; Li, W. et al., 2025), and the conditions under which mixing wins — measurable cognitive diversity (Serra, 2026b) and per-domain expertise asymmetry (Sakana AI, 2026) — are narrow, checkable, and usually unchecked. We formalise expertise routing as a shrinkage estimator over a prior seeded from published per-benchmark leaders and updated by measured outcomes, prove the resulting orchestration is bounded in agents, concurrency and tokens, and specify the falsification experiment we have not yet run. The practical claim is modest and, we think, correct: orchestration is a *scheduling* discipline, not an intelligence multiplier, and its returns are governed by the structure of the task graph and the measured heterogeneity of the pool — both of which a harness can compute before spending a token.

Status: SKETCH v0.2 (2026-09-03). v0.2 adds a **third axis** — *supply mechanisms* — and the five mechanisms that live on it (§4.5), a shadow-price formalisation of inventory levelling (§8.5), a rewritten routing algorithm (Annex A), and the first board-level measurement of the problem the axis exists to solve (§13). v0.1 was 2026-07-25.

Status: SKETCH v0.1 (2026-07-25). Codename **MAESTRO** (**M**odel-**A**ware **E**xpertise **S**election and **T**ask **R**outing **O**rchestration) is proposed and verified free in the J-series registry (J18 = STRIATUM is the highest used); **pending Oscar's sign-off before registering J19**. Brain analogy: the **thalamus** — the relay that

routes each incoming signal to the cortical area equipped to process it, and the closest anatomical match to a router that owns *dispatch* rather than *reasoning*.

Honest framing up front: this paper contains **one measured artifact of our own** (a routing plan, §6.3) and **no end-to-end outcome data**. Its contribution is a taxonomy, a cost model, and a falsification plan — not results. Every number is labelled *measured* or *projected*.

1. Introduction

A harness is the machinery around a model: the loop that plans, calls tools, spawns helpers, verifies, and retries. When practitioners say orchestration made their harness better, they usually mean one of two unrelated things. Either the same work finished sooner, or work that previously failed now succeeds. These are different axes. They have different cost structures, different failure modes, and — this is the part that gets lost — they frequently trade against each other. Adding a verification pass makes a harness smarter and slower. Splitting a refactor across eight agents makes it faster and, if the units were not truly independent, dumber.

The confusion is not merely terminological. It produces a specific, common, expensive mistake: a builder observes that a multi-agent configuration improved a benchmark, concludes that "more agents is better," and applies fan-out to tasks whose structure cannot benefit — paying a multiple of the token bill for a result the single best model would have produced alone. The literature already contains the correction. Li, J. et al. (2024) show accuracy scaling with sampled agents but explicitly bound the gain by *task difficulty*; Li, W. et al. (2025) show that Self-MoA — repeatedly sampling the single strongest model — often beats mixed Mixture-of-Agents, because mixing trades away average quality for diversity and the trade is not always favourable. The question is therefore never "should I orchestrate?" but "which mechanism, against which structural property of this task, at what price?"

This paper answers that question in three moves. §3 and §4 give the taxonomy, split by axis, each mechanism annotated with its precondition. §5 and §6 give the cost model and the boundedness argument a harness needs to run these mechanisms without unbounded spend. §7 states, as plainly as we can, the conditions under which orchestration does not pay — which we believe is the most useful section for a practitioner and the one most often omitted.

Scope and relation to sibling work. MAESTRO owns *who executes each step and how many run at once*. It does not own *which procedure to run* — that is the recipe substrate of PREFRONTAL (Serra, 2026c) — nor *how models should argue once convened*, which is the deliberation protocol of Round Table (Serra, 2026b). MAESTRO consumes both: it decides *whether* convening a Round Table is worth its cost on this particular unit, and it schedules the steps a PREFRONTAL recipe declares. Where those papers ask what deliberation and executive function *are*, this one asks what they *cost* and when to decline them.

2. Two axes, not one — and a third they both presuppose

Let a harness process a task decomposed into units $u_1 \dots u_N$. Write T for wall-clock latency, K for total tokens billed, and Q for solution quality on the task's own oracle.

- A **latency mechanism** reduces T with K approximately unchanged, by exploiting independence already present in the task graph. It cannot raise Q; at best it preserves it.
- A **capability mechanism** raises Q by increasing K. Nearly all of them also increase T, because the added computation sits on the critical path.

Two consequences follow immediately and are worth stating because they are routinely violated in practice.

(a) A latency mechanism applied to a dependent graph silently becomes a quality regression. Splitting units that share state across parallel agents does not merely fail to speed things up; it produces agents reasoning from stale or partial views of a tree that is changing underneath them. The wall-clock win is real and the correctness loss is invisible until integration.

(b) A capability mechanism applied to an easy task is pure cost. If the domain leader solves the unit alone, a three-model panel produces the same answer three times and pays a synthesiser to notice. This is the Self-MoA result (Li, W. et al., 2025) restated operationally: the diversity you are buying is only worth its price when the leader is *not* reliably right.

The discipline MAESTRO proposes is therefore to make the axis explicit at dispatch time — to require, for every orchestration decision, a named precondition that was actually checked.

And both axes presuppose a third. T, K and Q are all defined over calls that *happen*. A supply at its rate limit does not make the turn slower or dearer; it makes Q undefined, because there is no output to score. Section 4.5 adds the axis that governs whether the call happens at all, and §4.6 shows that it is not independent of the other two: every capability mechanism buys its quality with extra calls, and on a metered-by-clock portfolio each extra call is extra exposure to the limiter that ends the turn.

3. Latency mechanisms

Each entry states the mechanism, the structural precondition without which it cannot pay, and the resource it trades.

L1 — Parallel decomposition. Run independent units concurrently. Wall-clock becomes the slowest unit rather than the sum. *Precondition:* units are genuinely independent (disjoint writes). *Trades:* nothing but scheduling overhead, when the precondition holds. This is the single highest-value mechanism in the taxonomy and the only one that is close to free — which is precisely why its precondition deserves machine checking rather than authorial confidence.

L2 — Critical-path serialisation. When units are *mostly* independent but contend on a few shared resources, serialise only the contended resource rather than the whole run. In a coding harness this is a per-file lease: disjoint files proceed concurrently, shared files hand off. *Precondition:* contention is sparse relative to the unit set. *Trades:* handoff latency on contended items only. Amdahl's law is the governing bound — the serial fraction, not the agent count, determines the ceiling.

L3 — Intra-answer parallel generation. Decompose one *answer* into independently-generable parts, emit a skeleton, then fill the parts concurrently (Ning et al., 2023). *Precondition:* the answer is genuinely list-shaped, not a chain of dependent reasoning. *Trades:* coherence across parts; unsuitable for arguments where part k depends on part $k-1$.

L4 — Cascades. Attempt with a cheap model, escalate only on low confidence (Chen et al., 2023). *Precondition:* a usable confidence signal exists and the easy-case share is large. *Trades:* worst-case latency, which becomes cheap-attempt *plus* expensive-attempt. Cascades optimise the mean and punish the tail.

L5 — Right-sizing by routing. Send the task to the smallest model that can do it (Ong et al., 2024). *Precondition:* per-task difficulty is predictable before dispatch. *Trades:* quality when the difficulty estimate is wrong — the failure is silent, which makes calibration the whole game.

L6 — Speculative execution. Draft cheaply, verify expensively, keep the draft when it passes. The agent-level analogue of speculative decoding (Leviathan et al., 2023), which achieves 2–3× with *provably identical* outputs at the token level. *Precondition:* verification is substantially cheaper than generation — a condition that holds for token decoding and holds far

less obviously for agent work, where verifying a patch can cost as much as writing it. We flag this as the least-transferable mechanism in the taxonomy and decline to claim the decoding-level speedups carry over.

4. Capability mechanisms

C1 — Repeated sampling with a vote. Sample the same model n times, take the majority (Wang, X. et al., 2022; Li, J. et al., 2024). *Precondition:* errors are stochastic and uncorrelated across samples. *Trades:* $n\times$ tokens. This is the baseline every fancier mechanism must beat, and frequently is not beaten (Li, W. et al., 2025).

C2 — Cross-model ensembling and fusion. Rank and fuse candidate answers from different models (Jiang et al., 2023); or layer agents so each reads the previous layer's outputs (Wang, J. et al., 2024). *Precondition:* models make *decorrelated* errors — the quality-diversity trade must land on the diversity side. *Trades:* $n\times$ tokens plus a fusion step.

C3 — Debate with an adaptive aggregator. Models answer independently, then a synthesiser resolves disagreement (Du et al., 2023). The refinement that matters is *adaptive* chairing: Sakana AI (2026) reports the aggregator being chosen per query — GPT chairing a mathematical dispute, Gemini chairing a factual-recall one. *Precondition:* genuine disagreement exists *and* panellists are isolated. *Trades:* $n{+}1$ calls, serialised into two rounds.

C4 — Expertise routing. Send each unit to the model measured strongest in *that domain* rather than the strongest on average. This is Fugu's central mechanism and its most transferable finding (§8). *Precondition:* the pool exhibits per-domain asymmetry — if one model dominates everywhere, routing degenerates to "always use the best." *Trades:* essentially nothing at inference; the cost is the measurement infrastructure.

C5 — Verification and critique loops. Generate, critique, revise (Shinn et al., 2023; Madaan et al., 2023). The cross-provider variant — critic drawn from a *different* vendor than the builder — is the sharpest form, and Sakana AI (2026) documents it catching a concrete defect that the builder's own review did not. *Precondition:* the critic can detect the defect class the builder produces, which same-family critics are structurally worse placed to catch (§7.2). *Trades:* at least one extra round on the critical path.

C6 — Search over reasoning. Treat reasoning as a tree and search it (Yao et al., 2023). Sakana AI's AB-MCTS (Inoue et al., 2025) generalises repeated sampling by deciding adaptively whether to go *wider* (new attempt) or *deeper* (refine an attempt), and extends to a multi-model pool. *Precondition:* a scoring signal exists to guide the search. *Trades:* the most tokens of any mechanism here, by a wide margin.

C7 — Recursive self-call. The orchestrator reads its own prior output and decides whether to revise its coordination strategy, with recursion depth as a tunable inference-time parameter requiring no retraining (Sakana AI, 2026). *Precondition:* the orchestrator can recognise its own inadequate first attempt — a self-assessment problem, and therefore the mechanism most exposed to overconfidence. *Trades:* unbounded spend unless depth is capped, which is why §6 caps it.

4.1 The orthogonality table

Mechanism	T	K	Q	Precondition that must be *checked*
L1 parallel decomposition	$\downarrow\downarrow$	–	–	disjoint writes
L2 critical-path serialisation	$\downarrow$	–	–	sparse contention
L3 intra-answer parallel gen	$\downarrow$	–	$\sim$	answer is list-shaped
L4 cascade	$\downarrow$ mean, $\uparrow$ tail	$\downarrow$	$\sim$	confidence signal + easy-case mass
L5 right-sizing	$\downarrow$	$\downarrow\downarrow$	risk $\downarrow$	difficulty predictable pre-dispatch
L6 speculative execution	$\downarrow$?	$\uparrow$	–	verify $\ll$ generate (weak for agents)
C1 repeated sampling	$\uparrow$	$\uparrow\uparrow$	$\uparrow$	uncorrelated stochastic errors
C2 ensembling / fusion	$\uparrow$	$\uparrow\uparrow$	$\uparrow$?	decorrelated *across models*
C3 debate + adaptive chair	$\uparrow\uparrow$	$\uparrow\uparrow$	$\uparrow$	real disagreement + isolation
C4 expertise routing	–	–	$\uparrow$	per-domain asymmetry in the pool
C5 verify / critique	$\uparrow$	$\uparrow$	$\uparrow$	critic sees builder's blind spot
C6 tree search	$\uparrow\uparrow\uparrow$	$\uparrow\uparrow\uparrow$	$\uparrow\uparrow$	usable scoring signal
C7 recursive self-call	$\uparrow\uparrow$	$\uparrow\uparrow$	$\uparrow$?	orchestrator can judge itself
S1 feasibility veto	–	–	**0 $\rightarrow$ Q**	supply reachable job fits model will engage
S2 shadow pricing	–		–	windows publish utilization *and* a reset instant
S3 deadline burn	$\uparrow$	**$\uparrow\uparrow$ at price 0**	$\uparrow$	surplus is provably unconsumable before reset

Mechanism	T	K	Q	Precondition that must be *checked*
S4 failure-directed reroute	↑ on failure	↑ on failure	**0 → Q**	the error class is classifiable
S5 clock-aware fan-out	↓↓	–	–	supplies differ in *window shape*, not only in price

The column that matters is the last one. Every row is a claim that becomes false when its precondition fails, and most preconditions are *computable* — disjointness of writes, presence of disagreement, measured per-domain asymmetry. A harness that checks them before dispatch converts this table from advice into a policy.

Note what the Q column says for S1 and S4, and note that no L or C row says it. Every other mechanism moves quality by increments. The supply mechanisms move it between **zero and non-zero**, because a model that refuses the call produces nothing at all.

4.5 The third axis: supply mechanisms

Sections 3 and 4 share an assumption so basic that stating it feels pedantic: that every model in the pool is **reachable, at a stable price, whenever the orchestrator decides to call it.** That assumption is inherited from the literature's setting, where models are metered API endpoints billed per token, and it is the setting in which Fugu (Sakana AI, 2026), Self-MoA (Li, W. et al., 2025) and the debate line (Du et al., 2023) were all evaluated.

It is false for the class of harness this paper is actually about. A personal or small-team harness runs on **subscriptions**: a bundle of supplies, each with its own renewal clock, its own hard limiter, and — this is the part with consequences — a balance that is **destroyed at reset rather than carried forward.** Under that structure three things stop being true at once:

1. **Availability is not constant.** A supply at its limit is not "slower" or "dearer"; it is *absent*, and every routing table that ranks it is ranking a model that will not answer.
2. **Price is not the sticker.** A token from a bucket that will expire unspent cost *nothing*; a token from a bucket that is about to bind cost *everything downstream that now cannot run.* Two tokens with the same list price can differ in true price by the full width of that gap.
3. **Spending is not stateless.** Each call changes the feasible set for the rest of the window, which makes routing a scheduling problem over inventories rather than a sequence of independent argmaxes.

We call the mechanisms that address these **supply mechanisms** and label them S. They do not buy latency and they do not buy accuracy. They buy **continuity**: the property that the work happens at all, and that the portfolio is still able to do work tomorrow.

S1 — Feasibility veto. Rule a model out before comparing fitness, on three grounds that are constraints rather than preferences: *reachability* (the supply is spent, cooling after a failure, or unfunded), *capacity* (the job's token estimate exceeds the model's context window — a question of arithmetic, not of merit), and *engagement* (the family systematically declines legitimate work in this subject class, so the call returns a refusal at full price). *Precondition:* the supply publishes a usable quota signal, or the harness has observed enough failures to infer one. *Trades:* nothing — it removes candidates that would have produced zero.

S2 — Shadow pricing. Bend the cost axis by how far each supply is ahead of or behind an even consumption pace, so that fairness across supplies enters the existing selection **as a price rather than as a second rule** (§8.5). *Precondition:* windows publish both a utilization and a reset instant; without the reset the elapsed fraction is undefined and the price has no sign. *Trades:* nothing at inference; it re-orders an argmax the harness was already computing.

S3 — Deadline burn. When a subscription window is near reset carrying surplus that cannot physically be consumed at the current rate, the marginal price of those tokens is zero *by deadline*, and the correct response is not merely to spend but to **explore** — to run the composition modes, model pairings and strategies whose expected value never justified their price before. *Precondition:* the surplus is provably unconsumable (near-reset AND a real deficit; each half alone is a different and wrong state). *Trades:* nothing in the window; the risk is that a harness learns to treat the exception as the rule, which is why the mechanism must be armed by a predicate and not by a mood.

This is the only mechanism in the taxonomy whose token cost is **negative**, and it is also the only one that deliberately manufactures variance. Both facts follow from the same observation: a policy that never varies cannot be calibrated, and a burn window is the one moment when the price of finding that out is zero.

S4 — Failure-directed rerouting. Treat the route as an ordered chain rather than a point, and let the **error class choose the direction of the reroute** — a rate limit moves to a different *supply*, a context overflow moves to a strictly *larger window*, a refusal moves to a *different family*, a timeout moves to a *cheaper* rung. *Precondition:* failures are classifiable, which transport errors usually are and semantic ones often are not. *Trades:* one extra round on the failing turn, against the alternative of losing it.

The distinction between this and ordinary retry is the whole content of the mechanism. Retry asks "again?"; rerouting asks "*in which direction?*", and only the second one can recover from a failure whose cause is structural rather than transient. A chain that reroutes to a second rung on the same rate-limited supply survives nothing — supply diversity, not chain length, is the property that matters.

S5 — Clock-aware fan-out. A fan-out of N units costs $O(N)$ tokens at the leaves and $O(1)$ at the aggregator. It follows that the leaves should be drawn from the supply with the most headroom **and the most forgiving window shape**, while the chair may be the most expensive model on the board, because it is one call. *Precondition:* supplies differ in window *shape* and not merely in price. *Trades:* leaf quality against continuity, bounded by the fact that leaf work in a fan-out is by construction the decomposable part.

The shape term is doing the work here and it is easy to miss. Two supplies with identical utilization are not interchangeable if one publishes a five-hour limiter and the other publishes only a weekly one: a twenty-way burst on the first can trip a short window that the rest of the day's interactive work depends on, while the same burst on the second is invisible against a seven-day denominator. **Width should be allocated against window shape; depth against fitness.** We believe this is the most transferable idea in the section, and it is one that could not arise in a metered-API setting, because there are no clocks there to be shaped.

4.6 The interaction the two-axis model cannot express

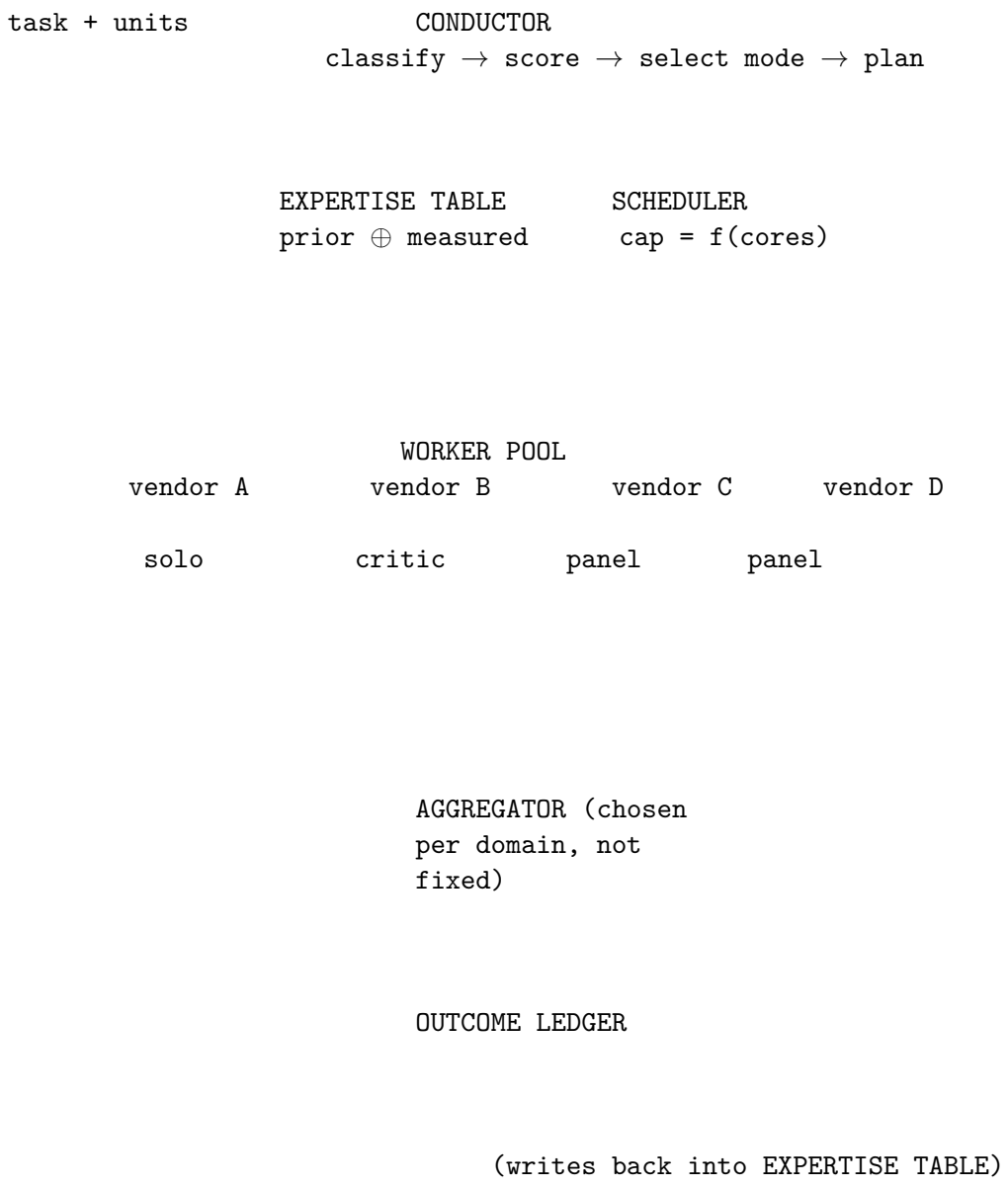
Capability mechanisms multiply supply risk. C1 at n=5, C3 with a three-model panel, and C6's tree search all buy quality by making more calls, and *each additional call is additional exposure to a limiter*. On a metered pool that is purely a money cost and the two-axis model prices it correctly. On a subscription portfolio it is also a **continuity** cost, and it is convex: the calls that push a supply over its limit do not merely cost their own price, they remove that supply from every subsequent turn in the window.

The practical consequence inverts a piece of standard advice. "Use a panel when the domain is contested" is sound in isolation; on a constrained portfolio it must become "use a panel when

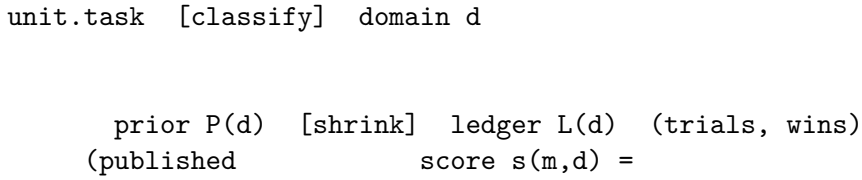
the domain is contested **and** the panel's supplies have the headroom to absorb it", and the second clause is checkable. This is why our own implementation gates composition on the operator's quality dial and opens it unconditionally only inside a burn window (§4.5, S3) — the one state in which the extra calls have no continuity cost at all.

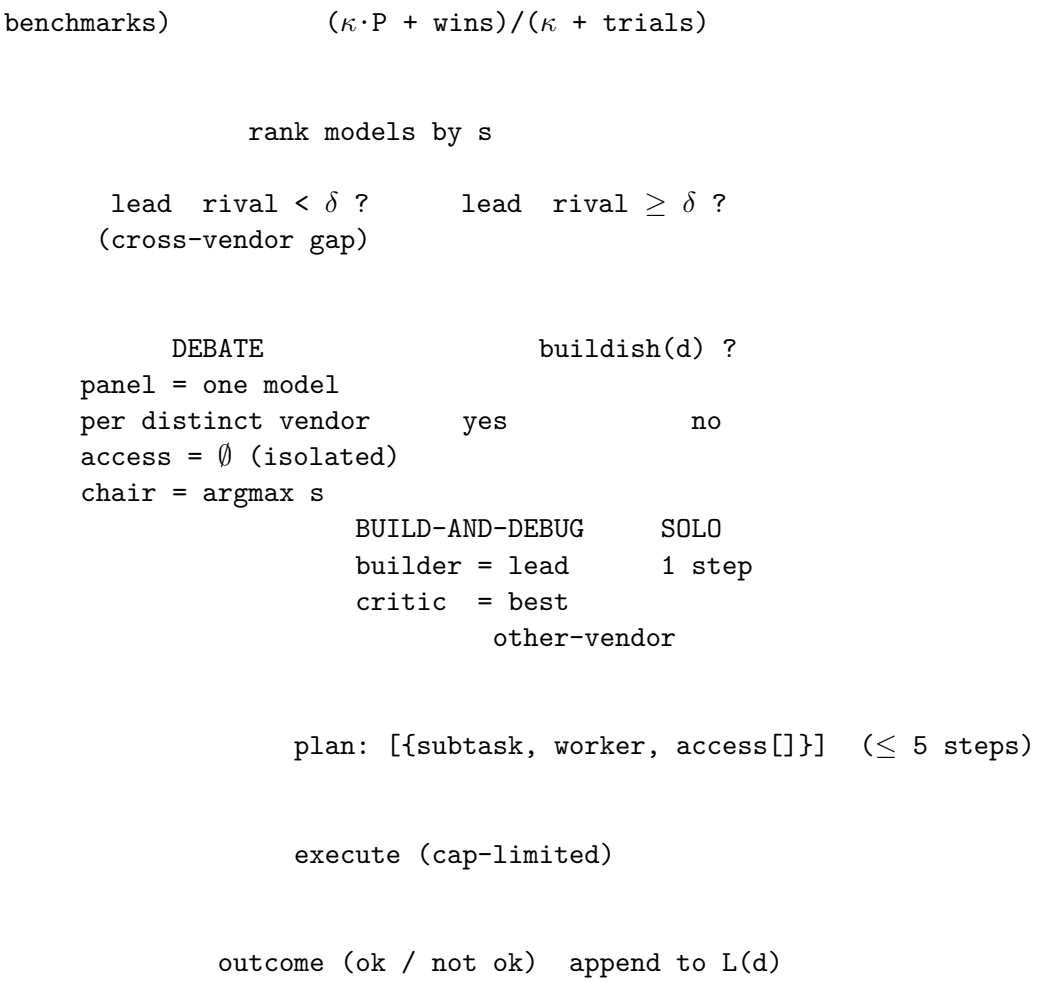
5. Architecture

5.1 System diagram



5.2 Data-flow diagram





The **access list** on each step is load-bearing rather than decorative: it is what enforces the isolation precondition of C3. An agent sees a prior step's output *only* if that step's index appears in its access list. Without it, a panel degenerates into a chain and the diversity being paid for is destroyed by the first answer.

6. Cost model and boundedness

6.1 Latency

For a run of N units under L1 with concurrency cap c, plan step-count s_i per unit, and mean per-agent latency $\bar{t}$:

$$T_{\text{parallel}} \approx \bar{t} \cdot \max_i s_i \cdot \left\lceil \frac{\sum_i s_i}{c \cdot \max_i s_i} \right\rceil + T_{\text{route}}$$

against a serial baseline of $\bar{t} \sum_i s_i$. Two readings matter. When $\sum s_i \leq c$, the run costs the *depth of the deepest plan*, not the total work — the ideal case. When $\sum s_i \gg c$, the cap dominates and additional decomposition buys nothing; the harness is throughput-bound and should stop fanning out.

6.2 Tokens

With C the mean tokens per agent call:

- baseline (one agent per unit): $K_0 = N\,C$
- MAESTRO ultra tier: $K \le (\sum_i s_i)\,C \le 5NC$ by the step cap (§6.4)

So the ceiling is a **5× token multiplier**, and the realised multiplier is the mean plan depth. This is the number a builder should hold in mind: composed orchestration is not marginally more expensive, it is a small-integer multiple, and it must buy a correspondingly large quality gain to be rational.

6.3 What we have actually measured

Exactly one artifact, and we state its limits precisely. Running our Conductor over a synthetic five-unit workload (a UI panel, a crash fix, a CI packaging failure, a literature summary, a README) on an 8-core host produced — **measured, 2026-07-25** — a plan of **18 agents across 3 vendors**, i.e. a mean plan depth of 3.6 and a token multiplier of **3.6**× against the one-agent baseline, with a concurrency cap of **6** (= cores 2). The routing distribution reproduced the qualitative pattern Sakana AI reports for Fugu: systems work to GPT, science to Gemini, debugging to the SWE-Bench leader, long-horizon work to the long-context model.

What this is *not*: it is a **plan**, not an execution. We have no completion times, no accuracy deltas, and an **empty outcome ledger** — every routing decision above was made from the seeded prior, with zero measured trials. The 3.6× token multiplier is therefore a *measured cost with an entirely unmeasured benefit*. Any claim that this configuration is better than running the single best model five times would be, at this stage, unfounded. §10 specifies the experiment that would settle it.

6.4 Boundedness invariants

A harness that can spawn agents that spawn agents must prove it terminates. We state four invariants, all enforced structurally rather than by convention:

- **I1 — Step ceiling.** Every unit's plan has $s_i \le 5$ steps. Adopted from Sakana AI's reported Fugu-Ultra limit; deeper plans did not help them and the cap makes token spend linear in N.
- **I2 — Concurrency cap.** At most $\min(16, \text{cores} - 2)$ agents run at once, floor 1. Two cores are reserved so the host stays responsive; the constant 16 bounds thrash independently of core count.
- **I3 — Recursion prohibition.** The orchestration trigger is disabled inside a running orchestration. C7 (recursive self-call) is therefore depth-capped by construction rather than by the orchestrator's own judgement — the mechanism whose precondition we trust least is the one we refuse to leave unbounded.
- **I4 — Lease expiry.** A contended resource held by a crashed participant is reclaimable after a bounded TTL, so L2 cannot deadlock; the worst case is a bounded stall, not a wedge.

Together: agents per run $\le 5N$, concurrent agents $\le \min(16, \text{cores} - 2)$, tokens $\le 5NC$, and no unbounded wait. Resource usage is bounded at all times.

7. When orchestration does not pay

This is the section we would most want a reader to keep, and it is a summary of other people's negative results as much as our own reasoning.

7.1 The pool may not be heterogeneous enough. Li, W. et al. (2025) find Self-MoA — the single strongest model, sampled repeatedly — outperforming mixed Mixture-of-Agents on

several benchmarks, — reported at +6.6% over MoA on AlpacaEval 2.0 and +3.8% on average across MMLU, CRUX and MATH — and identify the mechanism: mixing admits weaker models and the average-quality loss can exceed the diversity gain. Round Table (Serra, 2026b) reaches a compatible conclusion from the other direction, reporting that default multi-agent debate rarely beats strong single-agent self-consistency *except* when cognitive diversity is high. The operational reading: **diversity is a precondition to be measured, not an assumption to be enjoyed**, and a harness that cannot measure it should default to the single best model.

7.2 Same-vendor panels buy less than they appear to. Two models from one lab share training lineage and therefore share correlated errors, so a panel that looks like three opinions may be one opinion with two echoes. The clearest evidence we have found is from an adjacent modality: the *Hidden Clones* study (2026) shows that vision-language models from the same architectural family share correlated errors that flat voting ignores, and that aggregating *within* families before voting *across* them recovers 18–26 percentage points on their hardest tier. We flag the transfer honestly — that result is about VLM ensembles, not LLM agent panels, and we have not reproduced it in our setting. It is the reason our routing deduplicates panels by *vendor* and measures contest against the best model of a *different* provider, but it is a motivating analogy, not proof.

7.3 The task may be too easy. Li, J. et al. (2024) find the gain from more agents correlated with task difficulty. Under a fixed orchestration policy this means the easy tail of a workload is pure waste. A README does not need a panel.

7.4 Coordination is not free. Every added step is a place to lose context, and the isolation that makes a panel valuable is the same isolation that denies each panellist the others' findings. Sakana AI's access-list design exists precisely because unrestricted visibility caused early agents' actions to constrain later ones. Orchestration adds failure modes that a single call does not have.

7.5 Verification may not be cheaper than generation. L6's transfer from speculative decoding rests on verify $\ll$ generate. For agent work this is often false. We decline to claim the 2–3× decoding speedups carry to the agent level, and we would treat any such claim without measurement as unsupported.

8. Expertise routing as shrinkage

Fugu's contribution that we believe transfers best is not its architecture but its *epistemology*: it does not hand-write which model is good at what, it **measures** it. Sakana AI (2026) runs every worker over queries, scores against references, and converts measured performance into a soft target the router is trained on; the learned distribution then reproduces the benchmark leaders per domain without anyone encoding them.

Most harnesses cannot train a router. But the *estimator* is separable from the training, and can be reconstructed cheaply. Let $P(m, d)$ be a prior for model m in domain d, seeded from published per-benchmark leaders, and let the outcome ledger hold $(\text{trials}, \text{wins})$ per pair. Then

$$s(m, d) = \frac{\kappa\, P(m, d) + \text{wins}(m, d)}{\kappa + \text{trials}(m, d)}$$

is a shrinkage estimator that behaves correctly at both ends: with no local evidence it returns the published prior, and as trials accumulate the local measurement dominates. The constant κ is the prior's weight in pseudo-trials — how many of our own observations it takes to overturn a benchmark. We use $\kappa = 3$, which makes a single lucky run unable to unseat SWE-Bench Pro while a dozen consistent outcomes can.

Two properties make this worth stating as a contribution rather than a detail. First, it is **honest about provenance**: a routing decision can always report whether it rests on borrowed benchmarks or local evidence, which is exactly the distinction a practitioner needs when a route

looks wrong. Second, it **degrades gracefully**: a harness with an empty ledger is a published-leaders router, which is strictly better than a coin flip and no worse than the hand-written table it replaces.

The mode selection built on top is a two-line policy. If the leader's margin over the best *different-vendor* model is below δ, the domain is contested and gets a panel; otherwise, coding-shaped work gets a cross-vendor critic and everything else runs solo. The margin test is the checkable form of §7.1's precondition.

8.5 Shadow pricing as inventory levelling

Expertise routing tells us *who is best*. It does not tell us *whose tokens we should be spending right now*, and on a subscription portfolio those are different questions with different answers.

Let supply p carry a window of length W_p with τ_p remaining and utilization $u_p \in [0, 1]$. Define the elapsed fraction $e_p = 1 - \tau_p / W_p$ and the **shadow price**

$$\sigma(p) \;=\; \mathrm{clamp}(u_p - e_p,\; -1,\; +1)$$

$\sigma > 0$ means the supply is being drained faster than it renews: its remaining tokens are scarce, and each one spent now is one unavailable later in the window. $\sigma < 0$ means the opposite, and it means something stronger than "we have room" — under a subscription, the unspent remainder is **destroyed at reset**, so those tokens have no future value to preserve. The effective cost of a rung drawn on p is then

$$c_{\text{eff}} \;=\; c \times (1 + \lambda\, \sigma(p))$$

with λ setting how hard the pace term bends the axis (we use $\lambda = 0.6$: a supply a full window ahead of pace prices at 1.6× sticker, one a full window behind at 0.4×).

Three properties make this the right shape rather than merely a workable one.

It needs no second selector. The Pareto frontier and the operator's quality dial run unchanged; they simply run on a truthful axis. Even consumption across supplies is not a rule competing with quality — it is a consequence of pricing quality correctly. A harness that instead adds a fairness *rule* acquires two objectives that must be traded off by a constant nobody can calibrate.

It is self-correcting and cannot oscillate hard. As a supply is drawn down it becomes dearer and the frontier drifts toward its rivals; as the rivals catch up the prices converge. σ is continuous and bounded, so the feedback is proportional rather than bang-bang.

It gives the metered case for free. For a metered supply — a raw API key, a credit balance — unspent balance does *not* expire, so there is no deadline to be behind and σ is clamped to $\sigma \geq 0$: scarcity can still make it dearer, but nothing can make it cheap. A standing operational preference of the form "use the credit balance only when nothing else will do" is then not a special case in the router at all; it is what the price model already says.

The mechanism degrades honestly. A supply whose window publishes no reset instant has an undefined e_p and therefore no shadow price, and is priced at sticker — **unknown is never a discount**, because a discount on an unmeasured supply is precisely how a router talks itself into an exhausted one.

8.6 The burn predicate

$\sigma < 0$ says a supply is behind pace. It does not say the surplus is *unrecoverable*, and the difference matters, because acting on the first is over-eager and acting on the second is free. Burn arms only when both halves hold:

$$\text{burn}(p) \iff \underbrace{\tau_p < \theta W_p}_{\text{near reset}} \wedge \underbrace{e_p - u_p > d_{\min}}_{\text{real surplus}} \wedge p \text{ is a subscription}$$

with $\theta = 0.15$ and $d_{\min} = 0.25$. Each conjunct excludes a distinct error. *Near* alone fires at the end of every window including the ones we already emptied — nothing to save and nothing to spend. *Surplus* alone fires at the start of every window, where "behind pace" only means the week has not happened yet. And the subscription clause is what stops the mechanism from ever spending real money on the theory that it was about to expire.

9. Related work

Orchestration as a trained artifact. Sakana AI's Fugu (2026) is the closest and most important reference: an orchestrator trained to route, delegate, verify and synthesise over a pool of frontier models, reported at 73.7 on SWE-Bench Pro, 82.1 on Terminal-Bench 2.1, 95.5 on GPQA-Diamond and 90.8 on LiveCodeBench Pro — in each case at or above every individual model in its own pool. Fugu selects a single worker per query for latency; Fugu-Ultra composes multi-step workflows of at most five steps with per-step access lists. Its case studies contribute the two composed patterns we adopt (adaptive-chair debate, cross-vendor build-and-debug) and its Figure 5 is the empirical statement of per-domain asymmetry that C4 depends on. We take its *method* — measure, then route — and reconstruct it as a shrinkage estimator (§8) for harnesses that cannot train a router.

Routing and cascades for cost. FrugalGPT (Chen et al., 2023) established LLM cascades; RouteLLM (Ong et al., 2024) learns routing from preference data. Both optimise cost at fixed quality — the L4/L5 axis — and are complementary to expertise routing, which optimises quality at fixed cost.

Ensembling and deliberation. Self-consistency (Wang, X. et al., 2022) and Agent Forest (Li, J. et al., 2024) establish the sampling-and-voting baseline; LLM-Blender (Jiang et al., 2023) adds pairwise ranking and generative fusion; Mixture-of-Agents (Wang, J. et al., 2024) layers agents so each reads the previous layer. Multiagent debate (Du et al., 2023) is the origin of C3. The critical counterweight is Li, W. et al. (2025), whose Self-MoA result is the sharpest statement that mixing is not automatically better.

Search and inference-time scaling. Tree-of-Thoughts (Yao et al., 2023) frames reasoning as search; AB-MCTS (Inoue et al., 2025) generalises repeated sampling with an adaptive wider-or-deeper decision and extends to multi-model pools. These are the highest-cost, highest-ceiling members of the capability axis.

Self-correction. Reflexion (Shinn et al., 2023) and Self-Refine (Madaan et al., 2023) establish generate-critique-revise; the cross-vendor variant in C5 is the multi-provider specialisation.

Latency. Speculative decoding (Leviathan et al., 2023) gives 2–3× with identical outputs at the token level; Skeleton-of-Thought (Ning et al., 2023) parallelises generation of list-shaped answers. These are the two clean latency results in the literature, and the boundary of their transfer to agent-level work is discussed in §7.5.

Multi-agent frameworks. AutoGen (Wu et al., 2023) and comparable frameworks provide the conversational substrate for multi-agent systems; they are orthogonal to this paper, which is about the *policy* deciding what to run rather than the plumbing that runs it.

Sibling J-series work. Round Table (Serra, 2026b) supplies the deliberation protocol and the diversity-measurement apparatus MAESTRO's §7.1 precondition requires; PREFRONTAL (Serra, 2026c) supplies the recipe substrate whose steps MAESTRO schedules; Fractal Reasoning (Serra, 2026a) supplies the reflection loop that produces the outcome signals §8's ledger consumes.

10. Evaluation plan (not yet run)

The taxonomy is testable and we specify the falsification rather than deferring it. The comparison must be against the *right* baseline, which §7.1 tells us is not "one call" but **Self-MoA at matched token budget** — the strongest single model, sampled to the same spend.

- **H1 (routing).** Expertise routing beats the best single model at matched token budget on a mixed-domain workload. *Falsified if* the single best model at 5× sampling matches it.
- **H2 (asymmetry).** The gain from H1 is proportional to measured per-domain asymmetry in the pool. *Falsified if* the gain persists on a pool where one model dominates every domain — which would mean the gain came from extra sampling, not from routing.
- **H3 (cross-vendor critique).** A different-vendor critic catches defect classes a same-vendor critic misses. *Falsified if* same-vendor critique performs equally, which would remove the justification for vendor-deduplicated panels.
- **H4 (isolation).** Panels with empty access lists outperform panels that see each other. *Falsified if* shared-context panels match them — which would make the isolation machinery needless complexity.
- **H5 (parallel safety).** L1 on a verified-disjoint unit set produces the same final quality as serial execution. *Falsified if* parallel runs show elevated integration failures, which would mean the disjointness check is insufficient.

Instrumentation is the ledger of §8, which already records (domain, model, mode, outcome) per unit. The measurement gap is an *oracle*: coding units need a pass/fail signal, and until integration verification is wired as that oracle, the ledger records routing decisions without their outcomes. **Closing that loop is the precondition for every claim in this paper**, and is where v0.2 should begin.

11. Limitations

- **No end-to-end results.** We measure a plan, not an outcome (§6.3). Nothing here demonstrates that MAESTRO's orchestration beats a single strong model.
- **Domain classification is the weak link.** Fugu reads the query with a trained backbone; our classifier is keyword heuristics over the task text. A misclassified unit routes confidently to the wrong specialist, and the confidence is unwarranted. This is the most likely source of silent quality loss in the whole design.
- **The prior is borrowed.** Seeding from published benchmarks imports their generalisation gap: SWE-Bench Pro standing is not the same thing as being good at *our* debugging.
- **Vendor is a crude proxy for decorrelation.** We deduplicate panels by provider on the theory that shared lineage implies shared blind spots. It is a plausible proxy supported by the family- bias literature, not a measurement of the error correlation that actually matters.
- **Single-host scheduling.** The concurrency model assumes one machine's cores. Distributed orchestration changes the cap's meaning and is out of scope.
- **Latency is modelled, not profiled.** §6.1 is an analytic model with no wall-clock validation.
- **The shadow price assumes an even pace is the right pace.** $\sigma = u - e$ treats a linear burn as the reference trajectory, which is a modelling choice and not a fact about how work arrives. Demand is bursty and operator-driven; a harness that is "behind pace" on Monday may be perfectly scheduled for a Thursday deadline it knows nothing about. We accept this because the term is bounded and continuous — it biases, it does not decide

— but a demand-aware reference trajectory would be strictly better and we have not built one.

- **Window length is inferred from a label.** No vendor publishes the duration of the window it meters you against; we infer it from strings like `5-hour`, `7-day`, `weekly`. An unrecognised label yields no elapsed fraction and therefore no shadow price, which is the safe failure — but a *mis*-recognised one silently inverts the sign of the price, and nothing would report it.
- **Supply telemetry is partial and asymmetric.** On our own board one supply (OpenRouter) publishes no quota signal at any layer, so it can never be *known* to be spent. Absence must be read as unknown rather than as headroom; the cost of getting that backwards is routing traffic straight at an exhausted provider, and the temptation to read silence as room is structural.
- **The engagement term starts empty and stays weak.** It is seeded from nothing and learns only from refusals we actually observe, which makes it honest and also makes it slow: a family that will decline the work is not known to decline it until it has declined it twice. We prefer that to a hand-written table of who is prudish about what, but the first two turns pay for it.
- **Fan-out leaf quality is asserted, not measured.** S5 argues that leaf work in a decomposed task is by construction the part that tolerates a weaker model. That is a plausible claim about decomposition, not a measurement, and it is the assumption most likely to be wrong in a way that shows up as quietly worse output rather than as an error.

12. Open questions

1. Does per-domain asymmetry survive as frontier models converge? If every model becomes good at everything, C4's precondition evaporates and routing collapses to right-sizing (L5).
2. Can the domain classifier be replaced by a cheap learned probe, and is that probe's error rate below the routing gain it enables?
3. Is there a principled stopping rule for C6/C7 — a way to know a harder mechanism will not help *before* paying for it?
4. What is the correct κ? We chose 3 by judgement; it should be fitted once the ledger has data.
5. What is the correct λ? The shadow price's weight sets how much intelligence the harness is willing to trade for supply balance, and 0.6 is a judgement exactly as $\kappa = 3$ was. It is fittable against an objective the operator can state — minimise end-of-window variance in utilization subject to a floor on delivered quality — and we have not fitted it.
6. Does burn-window exploration actually produce usable calibration data, or only expensive variance? S3's justification is that the value of information is unchanged while its price has gone to zero. That is an argument, not a result; the test is whether the outcomes recorded in burn windows change any later routing decision.
7. Is *supply* diversity a good proxy for *error* diversity? We deduplicate panels by supply, which inherits v0.1's vendor-as-decorrelation-proxy limitation and adds a billing dimension to it — two models on one subscription may be from different houses entirely.
8. When does the third axis disappear? All five mechanisms exist because supplies are metered in currencies that expire. A harness on pure pay-as-you-go API keys has $\sigma \geq 0$ everywhere, no burn windows, and no clock shapes to allocate width against — the axis collapses to S1 and S4 alone. The interesting question is which way the industry moves.

13. The board, measured

The third axis was not derived; it was read off a screen. On **2026-09-03 at 15:39 UTC** the harness's own budget poller reported the following, and every number here is *measured* from provider APIs rather than projected:

supply	windows	utilization	resets	shape
Claude (Max)	5-hour · 7-day	39% · **71%**	19:29Z · 15:59Z	subscription, two clocks
ChatGPT (Plus)	5-hour · weekly	**100%** · **92%**	16:37Z · +4d	subscription, **spent**
GitHub Copilot	monthly premium	**100%**	month end	subscription, **spent**
xAI (Grok)	**weekly only**	**28%**	+4d	subscription, **no short window**
Gemini	rpm / rpd	0%	daily	free tier
OpenRouter	*none published*	unknown	—	metered

Three readings, and each one motivates a mechanism rather than illustrating it.

Two of six supplies were at 100% and a third was at 71%, while the supply with the most headroom sat idle at 28%. No fitness table can express this and no capability mechanism can repair it: the spent supplies were not worse that afternoon, they were *absent*. This is S1 and S2.

The idle supply was also the one with the friendliest window shape. xAI publishes a weekly window and no five-hour window at all, which means a wide burst against it cannot trip a short limiter — while the identical burst against Claude would consume the five-hour bucket that the rest of the day's interactive work depends on. The best place to put twenty concurrent leaves was therefore determined by *clock structure*, not by benchmark standing. This is S5.

And the harness's configured fallback list was empty. `agents.defaults.model.fallbacks = []`, with a complete, tested, working failover machinery sitting underneath it receiving nothing. Every rate limit that afternoon ended a turn that four other supplies could have served. This is S4, and it is the finding we would most like other harness authors to check in their own configuration, because the failure is invisible in code review: the machinery is present, the tests pass, and the input is empty.

We record the board rather than a summary of it because the distribution is the point. A harness that reports mean utilization across supplies would have shown roughly 56% that afternoon — a comfortable-looking number describing a portfolio in which a third of the supplies could not answer at all.

Annex A — routing algorithm (pseudocode)

v0.2 replaces v0.1's single-value `route()` with `plan()`. Three things changed and each is a mechanism from §4.5: the pool is filtered by feasibility before fitness is compared (S1), cost is read on the shadow-priced axis (S2), and the return value is an ordered **chain** rather than a point (S4). The expertise and composition logic is v0.1's, unchanged.

```
plan(unit, pool, ledger, supplies, quality, now):
```

```
    #  classify
    d     ← classify_domain(unit.task)               # keyword heuristic; §11
    j     ← classify_subject(unit.task)              # engagement class; S1
    k     ← estimate_tokens(unit)

    #  S1: feasibility BEFORE fitness - a constraint is not a preference
    feasible ← [ m in pool :
                    not spent(supply(m), now)
                and not cooling(supply(m), now)
                and funded(supply(m))
                and k ≤ context_window(m) · 0.8
                and refusals(family(m), j) < 2 ]
    if feasible = ∅: return NO_PLAN                  # say so; do not route anyway

    #  S2: price the axis truthfully
    for m in feasible:
        σ ← clamp(used(supply(m))  elapsed(supply(m)), 1, +1)
        if metered(supply(m)): σ ← max(0, σ)         # money never expires
        c_eff(m) ← c(m) · (1 + λ·σ)

    #  fitness: shrinkage over the ledger (§8), unchanged from v0.1
    s(m) ← (κ·prior[d][m] + wins[d][m]) / (κ + trials[d][m])

    #  the dial, anchored on a REFERENCE model, with a reserved set
    F      ← pareto_frontier(feasible on (c_eff, s))
    anchor ← best rung of reference_model in F        # "balanced" means THIS model
    burn   ← ∃ p : burn(p)                            # §8.6
    open_reserved ← (quality = max)  burn  (F ⊆ reserved)
    lead   ← dial_pick(F, quality, anchor, open_reserved)

    #  composition (§4): a capability mechanism is gated, not default
    mode ← solo
    if width(unit) > 1:                              # S5: clock-aware fan-out
        leaf ← argmax_p (headroom(p)  penalty_short_window(p))
        mode ← fan_out(leaves = best(leaf), chair = lead)
    else if quality ≥ deep  burn:
        rival ← best m in F with supply(m) ≠ supply(lead)
        if rival ≠ ∅  s(lead)  s(rival) < δ:         # contested → panel
            mode ← debate(panel = one per supply, ≤3, chair = lead, access = ∅)
        else if d ∈ {code, agentic}:
            mode ← build_debug(builder = lead, critic = rival)

    #  S4: the chain - ONE rung per OTHER supply, none below the dial's floor
    chain ← [ best rung of p : p ≠ supply(lead), rung ∈ feasible, s ≥ floor(quality) ]
             sorted by c_eff, truncated to 4

    return PLAN(lead, mode, chain, reasons)


on_failure(plan, error):                             # S4: direction, not repetition
```

```
match classify(error):
    rate_limit, overloaded → next supply in chain, penalise this one for a TTL
    capacity               → chain re-sorted by context window, descending
    engagement             → chain re-sorted by families with no refusal in j
    timeout                → chain re-sorted by c_eff, ascending
```

Invariants: every branch returns a plan of at most 5 steps (I1), so tokens $\leq 5NC$; the chain holds at most one rung per supply, so its length is bounded by the portfolio and not by the failure count; and `NO_PLAN` is a terminal state that reports itself, never a silent fallback to an infeasible model.

References

Chen, L., Zaharia, M., & Zou, J. (2023). *FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance.* arXiv:2305.05176.

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., & Mordatch, I. (2023). *Improving Factuality and Reasoning in Language Models through Multiagent Debate.* arXiv:2305.14325.

Inoue, Y., Misaki, K., Imajuku, Y., Kuroki, S., Nakamura, T., & Akiba, T. (2025). *Wider or Deeper? Scaling LLM Inference-Time Compute with Adaptive Branching Tree Search.* arXiv:2503.04412.

Jiang, D., Ren, X., & Lin, B. Y. (2023). *LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion.* arXiv:2306.02561.

Leviathan, Y., Kalman, M., & Matias, Y. (2023). *Fast Inference from Transformers via Speculative Decoding.* arXiv:2211.17192.

Li, J., Zhang, Q., Yu, Y., Fu, Q., & Ye, D. (2024). *More Agents Is All You Need.* arXiv:2402.05120.

Li, W., Lin, Y., Xia, M., & Jin, C. (2025). *Rethinking Mixture-of-Agents: Is Mixing Different Large Language Models Beneficial?* arXiv:2502.00674.

Madaan, A., et al. (2023). *Self-Refine: Iterative Refinement with Self-Feedback.* arXiv:2303.17651.

Ning, X., Lin, Z., Zhou, Z., Yang, H., & Wang, Y. (2023). *Skeleton-of-Thought: Prompting LLMs for Efficient Parallel Generation.* arXiv:2307.15337.

Ong, I., et al. (2024). *RouteLLM: Learning to Route LLMs with Preference Data.* arXiv:2406.18665.

Sakana AI (2026). *Sakana Fugu Technical Report.* arXiv:2606.21228.

Serra, O. (2026a). *Fractal Reasoning.* J-series J3.

Serra, O. (2026b). *Round Table: Exploiting Cognitive Diversity as Computational Resource in Persistent AI Agents.* J-series J6.

Serra, O. (2026c). *PREFRONTAL: The Recipe Execution Substrate — How a Composable, Self-Authoring, Looping Workflow Engine Becomes an Agent's Executive Function.* J-series J13.

Shinn, N., Labash, B., & Gopinath, A. (2023). *Reflexion: An Autonomous Agent with Dynamic Memory and Self-Reflection.* arXiv:2303.11366.

Wang, J., Wang, J., Athiwaratkun, B., Zhang, C., & Zou, J. (2024). *Mixture-of-Agents Enhances Large Language Model Capabilities.* arXiv:2406.04692.

Wang, X., Wei, J., Schuurmans, D., et al. (2022). *Self-Consistency Improves Chain of Thought Reasoning in Language Models.* arXiv:2203.11171.

Wu, Q., Bansal, G., Zhang, J., et al. (2023). *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework.* arXiv:2308.08155.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023). *Tree of Thoughts: Deliberate Problem Solving with Large Language Models.* arXiv:2305.10601.

Hidden Clones: Exposing and Fixing Family Bias in Vision-Language Model Ensembles (2026). arXiv:2603.17111. (Cited by title; author list not verified at time of writing.)

References