TOKEN-CONSTRAINED AGENTIC AI: CAPACITY CURVES AND BUDGET-AWARE REASONING ALLOCATION

Anonymous authors
Paper under double-blind review

ABSTRACT

An agentic system is given a task and an allowance. Benchmarks score the first and ignore the second: a run that would have been correct with twice the tokens scores exactly like one that was affordable. We make the allowance part of the state. Writing an architecture as a *budget-augmented stochastic Petri net* with state (M, b), the workflow marking and the remaining budget, makes the constraint *structural*: a transition the budget cannot pay for is disabled, so its call is never attempted, and exhaustion joins success and error as a third absorbing outcome. The *capacity curve* $B^\star(\varepsilon)$, the least budget attaining reliability $1-\varepsilon$, becomes computable rather than sampled. Four results follow. Accuracy does not order missions: a strictly more accurate reasoner is strictly worse over an interval of budgets *exactly as wide as its price premium*, so the inversion is generic, not a counterexample. Every fixed configuration has a reliability ceiling no budget crosses. Two capacity curves must cross whenever the higher-scoring configuration starts paying off later, so a score-ordered leaderboard is budget-invariant only under a condition no benchmark reports. And because spending is irreversible the decision process is acyclic, so the optimal budget-aware policy is one backward pass.

We then test the claims on $8{,}190$ published agent trajectories. Reconstructing what each run spent turns every leaderboard entry into a capacity curve, and the ordering changes with the allowance: the top-scoring configuration has zero observed success below $23{,}126$ tokens (exact paired test against a lower-ranked one, $p < 10^{-16}$) and 41 pairs cross. Selecting on capacity rather than score reaches the same reliability with 88% fewer tokens, and 45.7% of that benchmark's recorded failures turn out to be runs that produced no answer at all.

1 INTRODUCTION

An agentic system is given a task and an allowance. Benchmarks score the first (Mialon et al., 2024; Jimenez et al., 2024; Liu et al., 2024; Yao et al., 2025). They do not ask whether the agent could have *afforded* the answer, nor what it should have spent where. The question has sharpened as models have gained the ability to spend variable effort: an agent can think longer, call a larger model, or run an extra verification pass, and the prices interact, because budget spent reasoning is budget unavailable for the tool call the reasoning was for. What draws the allowance down is concrete: the reasoning trace, the arguments and (usually much larger) the *results* of every tool call as they re-enter the context, retrieved passages, inter-agent messages, critic passes, and retries that pay twice for one step. None is a fixed cost of the task; each is a decision, and the decisions compete for one pool. On real logs that pool is spent lopsidedly: completion tokens, the thing a thinking-budget parameter buys, are 1.40% of the bill.

The failure this creates. Consider a mission that observes, reasons, verifies and acts, with three reasoning levels correct 70%, 90% and 98% of the time at costs 2, 5 and 10 units, a budget of 14 and surrounding steps costing 6. Choosing the most accurate level is locally optimal by every metric an evaluation reports. It is also fatal: it consumes 10 of the 14 units, leaving too little for the verification and action it was meant to enable, and the mission succeeds with probability 0.000

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

against 0.900 for the 90% level. Each level performs exactly as advertised; the best one destroys the mission. Theorem 1 shows this is not a constructed anomaly: in *any* architecture with a priced menu of reasoning levels and a tail of work to pay for afterwards, the accuracy and mission orderings disagree on a set of budgets whose width is exactly the menu's price spread. One could discover this by running the configuration, and for one configuration one should. But a builder's questions are counterfactual and combinatorial, and answering them by deployment means paying for the failures.

Contributions.

(i) **Budget as state** (§2): a budget-augmented stochastic Petri net with state (M, b) whose transitions are gated by $b \geq c_t$, so the constraint is structural rather than an expected-cost side condition, and token exhaustion becomes a third absorbing outcome beside success and error.

(ii) **Capacity metrics with an estimator** (§3): the capacity curve $B^\star(\varepsilon)$ and the exhaustion probability η, which separates "the agent was wrong" from "the agent ran out". The measured capacity curve is an empirical sub-distribution function, so one DKW band covers it uniformly and inverts to a confidence interval on $B^\star$ (Prop. 1).

(iii) **Four results** (§4): the inversion is generic, its interval as wide as the price premium (Thm. 1); a reliability ceiling bounds every fixed configuration independently of budget (Thm. 2); two capacity curves must cross whenever the higher-ceiling one has the later *onset*, so agreement of the ceiling and onset orderings is necessary for a score ranking to hold at every budget (Thm. 3); and budget monotonicity makes the decision process acyclic, so the optimal policy is one backward pass dominating every fixed level by a margin we bound (Thms. 4 and 5).

(iv) **The claims on independently published runs** (§6): $8{,}190$ AppWorld trajectories read for what they cost rather than what they did, giving measured capacity curves with uniform bands, 41 crossing pairs, an 88% token saving from selecting on capacity instead of score, 45.7% of recorded failures identified as non-conclusions, and the model's own independence assumption tested and found to err toward over-provisioning.

2 Making the budget part of the state

The formal scope is a *cumulative, non-renewable, per-mission inference budget* in tokens or the money they cost, which is what a per-task spend ceiling and a per-mission token allowance both are, and deliberately not a context window (occupancy, freeable by truncation), a rate limit (replenishes), or a per-call output cap; those are covered only where they reduce to it, as a context window does through the cost of the calls it permits.

The augmentation. We track not only which step the mission has reached but how much budget is left, and forbid any step the budget cannot pay for. Let a stochastic place/transition net have marking M, incidence matrix C and rates $\lambda_t(M)$, and give each transition a *token cost* $c_t \in \mathbb{N}$, zero when it is not a model call; the state becomes (M, b) with $b \in \{0, \dots, B\}$ the remaining budget, started at (M_0, B). Write

$$T_{\mathrm{En}}(M,b) = \{t : \ M \geq \mathit{Pre}(:,t) \ \text{ and } \ b \geq c_t\}, \qquad q_{x,y} = \sum_{\substack{t \in T_{\mathrm{En}}(M,b) \\ (M + C(:,t),\, b - c_t) = y}} \lambda_t(M,b) \qquad (1)$$

where the first is the set of transitions that are both workflow-enabled and affordable and the second is the generator on distinct states $x = (M, b)$ and y, with $q_{x,x} = -\sum_{y \neq x} q_{x,y}$. The mission moves from "step M with b left" to "step $M + C(:,t)$ with $b - c_t$ left" at the rate of t, and only if t is affordable. That second condition is the whole point: an unaffordable transition is disabled, so the call is never attempted. It is absent from the state space rather than present and refused, so the constraint holds under *every* assignment of rates, rather than in expectation as under a constrained MDP (Appendix G).

Definition 1 (Budget-augmented stochastic Petri net)**.** A *BSPN* is $\mathfrak{B} = (P, T, \mathit{Pre}, \mathit{Post}, \Lambda, c, M_0, B)$ with P, T finite disjoint sets of *places* and *transitions*; $\mathit{Pre}, \mathit{Post} \in \mathbb{N}^{|P| \times |T|}$ the arc multiplicity matrices and $C = \mathit{Post} - \mathit{Pre}$ the incidence matrix; Λ assigning each t a rate $\lambda_t(M, b) > 0$; $c : T \to \mathbb{N}$ a *token cost*; $M_0 \in \mathbb{N}^{|P|}$ the initial marking; $B \in \mathbb{N}$ the initial budget. Its state space is the reachable subset of $\mathbb{N}^{|P|} \times \{0, \dots, B\}$ from (M_0, B) under (1), with the generator above.

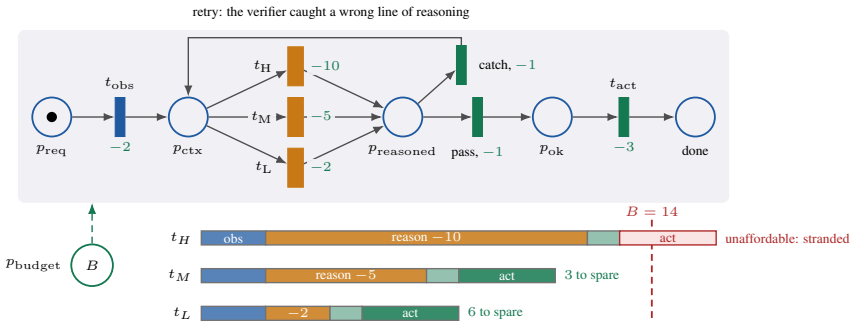

Figure 1: **The running mission as a token-aware net, and what each reasoning level leaves.** *Above*: the shaded row is the ordinary workflow. The verifier is two transitions in conflict on p_{reasoned}, *pass* and *catch*, since a probabilistic outcome is a choice between transitions rather than one transition with two outputs; the third branch, a wrong conclusion the verifier misses, is absorbed in F and is omitted here. The three reasoning transitions compete for the same context mark, so *choosing a reasoning level is choosing which of them fires*, and the green numbers are what this paper adds: each transition draws that many marks from p_{budget} and never returns them. *Below*: the same three choices as a ledger against $B = 14$. Only the reasoning segment differs, yet the deep level consumes so much that verification and action no longer fit. It is affordable, and affording it strands the mission, which is Theorem 1 at one point of its interval.

Colors (Jensen and Kristensen, 2009), age annotations and non-exponential timing are deliberately absent: the class is the stochastic place/transition nets of Molloy (1982) and Murata (1989) with one counter added, and no proof below uses anything that richer classes add, which is what lets Theorem 4 be about *every* model in it. Two processes live on the net and §4 needs both: under a *fixed* configuration the competing reasoning transitions are a stochastic race, while under a *policy* a router picks the tier and only then does anything stochastic happen, making the object a continuous-time MDP whose action set $A(M, b)$ is the maximal mutually non-competing subsets of $T_{\text{En}}(M, b)$ (Appendix B); σ_B^{π} is the absorption probability under π and $\pi^\star$ the maximizer.

Three outcomes, not two. Let Σ and F be the augmented states in which the mission has succeeded and erred. Everything else that cannot move is either *budget exhaustion*, $\mathcal{E} = \{(M, b) \notin \Sigma \cup F : T_{\text{En}}(M, 0) \neq \emptyset,\ T_{\text{En}}(M, b) = \emptyset\}$, in which the workflow has moves and the budget can pay for none, or an ordinary deadlock with $T_{\text{En}}(M, 0) = \emptyset$, which is a modeling error or a genuine dead end and is classified with F throughout. The distinction that matters operationally is between F and $\mathcal{E}$: a mission ending in F needs a better model, one ending in $\mathcal{E}$ a bigger allowance or a cheaper plan, and a success rate alone cannot tell them apart. §6 finds a benchmark on which the two are conflated and the split is close to even.

Stochastic consumption. An LLM call does not consume a fixed number of tokens. If call i consumes k units with probability $p_i(k)$, split its transition into branches of cost k and rate $\lambda_i p_i(k)$; if k units also buy correctness $q_i(k)$, split again into $\lambda^+_{i,k} = \lambda_i p_i(k) q_i(k)$ and $\lambda^-_{i,k} = \lambda_i p_i(k)[1 - q_i(k)]$, which charges for "more tokens, more reasoning, higher accuracy" and does *not* assume consumption and correctness are independent: $q_i(k)$ may vary freely with k, and §6 measures a corpus where it must. One convention matters. Under (1) a branch with $k > b$ is not enabled, so cost is drawn from p_i *conditioned on affordability*, which is right when the system can bound a call before making it; when cost is unknown until the call returns, that branch must instead be routed into $\mathcal{E}$: the call is made, the budget overrun, the mission stopped (Appendix K). Which one a deployment is in is an architectural fact, and it changes η.

Running mission. Observe, reason, verify, act (Figure 1), with the verifier catching a wrong line of reasoning with probability $v = 0.6$, after which reasoning is retried at further cost. Level L costs 2 at rate 1.0 and is correct with probability 0.70; M costs 5 at rate 0.667 and 0.90; H costs 10 at rate 0.4 and 0.98. The other steps, namely observe (2, rate 2.0), verify (1, rate 4.0) and act (3, rate 1.0), cost 6 in total, so with $B = 14$ level H is affordable but leaves the mission unfinishable. Appendix C maps

each agentic phenomenon (tier choice, reasoning effort, retries, variable-length generation, shared quotas) to a net construct and each construct to a chain quantity; Appendix I works four deployed architectures through it.

3 Capacity metrics, and how to estimate them

Write $\tau_\Sigma, \tau_\mathcal{E}$ for the hitting times of success and exhaustion, D for the deadline and $\tau = \tau_\Sigma \wedge \tau_F \wedge \tau_\mathcal{E}$, and define the *token-constrained success* σ_B, the *exhaustion probability* η, the *expected consumption* $\bar{C}$ and the *capacity curve* $B^\star$ by

$$\sigma_B(D) = \mathbb{P}_{(M_0,B)}\{\tau_\Sigma \le D,\ \tau_\Sigma < \tau_\mathcal{E}\}, \qquad \bar{C}(B,D) = \mathbb{E}\big[B - b_{\tau\wedge D}\big],$$
$$\eta(B,D) = \mathbb{P}_{(M_0,B)}\{\tau_\mathcal{E} \le D,\ \tau_\mathcal{E} < \tau_\Sigma\}, \qquad B^\star(D,\varepsilon) = \min\{B : \sigma_B(D) \ge 1-\varepsilon\}.$$

σ_B is what a deployment is judged on once cost is real; η is currently invisible, since a benchmark records a failure without recording that the agent ran out rather than erred, and those call for opposite interventions; and $B^\star$ is the strongest of the four, answering not "how well does this do with what it has" but "how much does it need". All four are absorption or transient quantities on (1), computed by a linear solve or by uniformization with B as a sweep parameter, and nothing is sampled. A capacity curve can also be *estimated* from logs, and that estimator has a structure making it testable rather than merely descriptive.

Proposition 1 (The measured capacity curve is an empirical CDF, and inverts)**.** *Run a fixed configuration on n tasks drawn i.i.d. from a task distribution and set Z_i = the run's cost if it succeeded, $+\infty$ otherwise. Then $\sigma(B) = \mathbb{P}(Z \le B)$ and $\hat{\sigma}(B) = n^{-1}\sum_i \mathbf{1}\{Z_i \le B\}$ is the empirical distribution function of Z, so by the Dvoretzky-Kiefer-Wolfowitz inequality with Massart's constant (Massart, 1990), with probability at least $1-\alpha$, $\sup_{B\ge 0}|\hat{\sigma}(B) - \sigma(B)| \le d_n(\alpha) := \sqrt{\frac{1}{2n}\log\frac{2}{\alpha}}$ simultaneously in B; and on the same event, simultaneously in ε, $\hat{B}^\star(\varepsilon + d_n) \le B^\star(\varepsilon) \le \hat{B}^\star(\varepsilon - d_n)$, where $\hat{B}^\star(u) := \min\{B : \hat{\sigma}(B) \ge 1-u\}$.*

The proof is three lines (Appendix J); what matters is that it makes the empirical object of §6 a statistic with a guarantee. At $n = 168$ and $\alpha = 0.05$ the band is $d_n = 0.105$ uniformly over the entire curve and 0.067 at $n = 417$, while pointwise the Clopper-Pearson interval is never wider than ± 0.078. Because every configuration ran *the same* tasks, differences between two curves are also paired and admit an exact McNemar test, which §6 reports for the headline crossing.

4 What the budget does to mission outcomes

4.1 Accuracy does not order missions, and the interval on which it fails has a known width

The introduction's example is arithmetic, and arithmetic invites the reply that it was constructed. It was not: the phenomenon is forced by two features every agentic architecture has, a menu whose more accurate options cost more and a tail of work still to be paid for afterwards. Fix a reachable marking M_r at which a menu L of controlled transitions is available, all leading to the same successor M_+ and differing only in cost c_ℓ, rate and correctness q_ℓ; let the *completion tail* $\kappa = \min\{\sum_{t\in u} c_t : u$ a firing sequence from M_+ to $\Sigma\} \in (0,\infty)$ be the cheapest way to finish once the reasoning is done. Write r for the budget remaining on arrival at M_r, and σ_r^ℓ for the probability of reaching Σ from (M_r, r) under the constant policy π_ℓ.

Theorem 1 (Inversion interval)**.** *Let $\ell, \ell' \in L$ with $c_\ell < c_{\ell'}$. Under the constant policies $\pi_\ell, \pi_{\ell'}$,*

$$\sigma_r^{\ell'} = 0 \ < \ \sigma_r^{\ell} \qquad \text{for every } r \text{ with } c_\ell + \kappa \ \le \ r \ < \ c_{\ell'} + \kappa.$$

This set is an interval of width exactly $c_{\ell'} - c_\ell$, and its width depends on no rate, no correctness probability and no property of the task distribution. In particular, if $c_1 < \cdots < c_m$ and $q_1 < \cdots < q_m$, the most accurate level returns exactly zero on $[0, c_m + \kappa)$ while some cheaper level is already positive on $[c_1 + \kappa,\ c_m + \kappa)$: the accuracy ordering and the mission ordering disagree over a range of budgets as wide as the menu's price spread.

Proof. Suppose $r < c_{\ell'} + \kappa$. If $r < c_{\ell'}$ then ℓ' is unaffordable, so by (1) $\pi_{\ell'}$ has no enabled controlled transition at M_r and Σ is unreachable. Otherwise $c_{\ell'} \le r < c_{\ell'} + \kappa$ and after ℓ' fires the residual is $r - c_{\ell'} < \kappa$; by definition of κ every firing sequence from M_+ to Σ costs at least κ, so none is executable under (1) and Σ is again unreachable. In both cases $\sigma_r^{\ell'} = 0$ exactly, not approximately. Conversely if $r \ge c_\ell + \kappa$ the residual after ℓ fires is at least κ, so a minimum-cost completing sequence is executable; being finite with all rates positive it is realized with positive probability, so $\sigma_r^{\ell} > 0$. Finally $\{r \in \mathbb{N} : c_\ell + \kappa \le r < c_{\ell'} + \kappa\}$ has $c_{\ell'} - c_\ell$ elements. □

The running mission is one point of this interval: $\kappa = 4$ and $c_{\mathrm{M}} = 5$, $c_{\mathrm{H}} = 10$, so the inversion holds for every $r \in [9, 14)$ and $B = 14$ leaves $r = 12$ after observation. The reading is sharper than "there exists a counterexample": selecting by benchmark accuracy is valid only above $c_m + \kappa$, the invalid region is as wide as the price spread, and both are known in advance.

4.2 Budget and reasoning quality are not substitutes

Theorem 2 (Reliability ceiling)**.** *Assume (C1) rates, costs and correctness probabilities do not depend on b; (C2) the policy is a constant π_ℓ; (C3) the* unconstrained *chain, obtained by deleting the affordability condition from* (1)*, is absorbed almost surely in $\Sigma \cup F$. Then $\sigma_B^\ell(\infty)$ is nondecreasing in B and $\sigma_B^\ell(\infty) \uparrow \bar\sigma_\ell = \mathbb{P}^{\mathrm{unc}}_{\pi_\ell}\{\tau_\Sigma < \tau_F\}$, a quantity of the workflow and the policy alone. If F is reached with positive probability in the unconstrained chain then $\bar\sigma_\ell < 1$, and for every target $\varepsilon < 1 - \bar\sigma_\ell$ the capacity curve $B^\star(\varepsilon)$ is* undefined*: no budget attains it.*

Sketch; Appendix J. Under (C1) and (C2) the budget is a pure feasibility filter: the state space at B embeds in that at $B' > B$ by $(M, b) \mapsto (M, b + B' - B)$, so a coupling on paths gives monotonicity, and the bounded monotone limit is the unconstrained absorption probability. It is (C1) that makes this a filter rather than a reweighting; under b-dependent rates a larger budget can change a race and reduce success, and the theorem is false. □

Corollary 1 (Closed form for a verify-retry loop)**.** *If a wrong conclusion is caught with probability $v < 1$ and retried, and otherwise absorbed in F, then $\bar\sigma_\ell = q_\ell + (1 - q_\ell)v\bar\sigma_\ell$, so $\bar\sigma_\ell = q_\ell / [1 - v(1 - q_\ell)] < 1$ whenever $q_\ell < 1$.*

Reliability is bought either by being right the first time or by being caught when wrong, so a level with $q_\ell = 0.7$ under a verifier catching 60% of errors cannot exceed 0.854 however long it keeps trying: a team targeting 98% cannot get there with the cheap model at any price. Buying tokens and buying reasoning quality are different purchases, the mission-level analogue of resampling against an imperfect verifier saturating (Stroebl et al., 2024).

4.3 When is a leaderboard's ordering the ordering under a budget?

Theorems 1 and 2 give each configuration two numbers: an *onset* $\beta_a = \inf\{B : \sigma_a(B) > 0\}$, the least allowance at which it delivers anything, and a *ceiling* $\bar\sigma_a = \sup_B \sigma_a(B)$, which is what a leaderboard publishes. Only one of the two is ever reported.

Theorem 3 (Crossing, and a necessary condition for budget-invariance)**.** *Let a, a' have nondecreasing capacity curves. If $\bar\sigma_a < \bar\sigma_{a'}$ and $\beta_a < \beta_{a'}$ then the curves cross: $\sigma_a(B) > 0 = \sigma_{a'}(B)$ for every $B \in (\beta_a, \beta_{a'})$, while $\sigma_a(B) < \sigma_{a'}(B)$ for all sufficiently large B. Consequently, for the ranking by $\bar\sigma$ to be the ranking at* every *budget it is necessary that β be nonincreasing in $\bar\sigma$ across the compared set: the higher-scoring configuration must also start paying off earlier. The condition is necessary and not sufficient, and the number of crossing pairs is at least $\#\{(a, a') : \bar\sigma_a < \bar\sigma_{a'},\ \beta_a < \beta_{a'}\}$.*

Sketch; Appendix J. On $(\beta_a, \beta_{a'})$ the onset gives $\sigma_a > 0 = \sigma_{a'}$, while for large B both curves are within any δ of their strictly ordered ceilings. Necessity is the contrapositive; insufficiency is a pair agreeing at both ends and crossing in the middle. □

The hypothesis is not exotic. Within a menu Theorem 1 identifies the onset exactly, $\beta_\ell = c_\ell + \kappa$ plus whatever was spent before M_r, so whenever accuracy is bought with money, which is the entire premise of a model tier list, it holds for *every* pair and crossing is a certainty. A leaderboard reporting

only $\bar{\sigma}$ reports one endpoint of an ordering it gives the reader no way to check; §6 measures both numbers for fourteen published configurations and finds 41 crossing pairs.

4.4 Optimal allocation is a single backward pass

Theorem 4 (Budget monotonicity makes the problem acyclic)**.** *Suppose every cycle in the underlying net contains a transition with $c_t \geq 1$ and the reachable marking set $\mathcal{M}$ is finite. Then the reachable augmented state set $S_B \subseteq \mathcal{M} \times \{0, \ldots, B\}$, with $|S_B| \leq (B+1)|\mathcal{M}|$, is a directed acyclic graph, and the policy maximizing $\sigma_B(\infty)$ is computed exactly by backward induction over it, with no convergence criterion, in time $O(|S_B||A|d)$ where $|A|$ bounds the actions at a state and d the successors of a state-action pair.*

Sketch; Appendix J. Every transition either leaves b unchanged or strictly decreases it and no cycle consists entirely of the former, so b is a strictly decreasing potential along any cycle, which therefore cannot exist. One detail matters for the algorithm: ordering states by b alone is *not* a topological order, since zero-cost transitions move within a layer, so the induction processes layers in increasing b and, inside each, in a topological order of its zero-cost subgraph. □

This is the structural payoff of putting the budget in the state: the feature that makes exhaustion expressible, irreversible spending, is what makes the optimization easy. Retries do not break it, since a retry loop revisits the same marking but never the same budget.

Theorem 5 (Adaptive dominance, with an explicit margin)**.** *Let $\pi^\star$ be the policy of Theorem 4 and π_ℓ any fixed level. Then $\sigma_B^{\pi^\star} \geq \sigma_B^{\pi_\ell}$ for every B and $B^\star_{\pi^\star}(\varepsilon) \leq \min_\ell B^\star_{\pi_\ell}(\varepsilon)$. Moreover, if at budget r level ℓ completes on a first attempt ($r \geq c_\ell + \kappa$) but not on a second after a detected error ($r - c_\ell - c_{\mathrm{ver}} < c_\ell + \kappa$), while a cheaper ℓ'' does ($r - c_\ell - c_{\mathrm{ver}} \geq c_{\ell''} + \kappa$), then $\sigma_r^{\pi^\star} - \sigma_r^{\pi_\ell} \geq (1 - q_\ell)\, v\, \sigma^{\ell''}_{r - c_\ell - c_{\mathrm{ver}}}$.*

Sketch; Appendix J. Each π_ℓ is feasible in the maximization. For the margin, $\pi^\star$ is at least as good as the policy playing ℓ first and ℓ'' on a detected retry, which earns the extra term because π_ℓ's own second attempt lands below $c_\ell + \kappa$ and returns 0 by Theorem 1. □

5 Exact analysis of the model

Nothing here is sampled: absorption probabilities come from the backward induction of Theorem 4 and are exact to floating point, and deadline-constrained quantities come from uniformization, truncated at a tolerance Appendix L bounds. Figure L2 is the result, and every feature of it is a theorem. There are three regimes and so two changes of ordering: below 11 units only the cheap level is affordable and returns 0.700 while both others return exactly 0; between 11 and 15 the mid level is best; from 16 the deep level is. The optimal policy is a lookup table over remaining budget (L on $[2, 8]$, M on $[9, 13]$, H from 14) whose thresholds are $c_{\mathrm{M}} + \kappa = 9$ and $c_{\mathrm{H}} + \kappa = 14$ from Theorem 1, and it says what a practitioner would not otherwise guess: reason cheaply when poor, because an expensive answer you cannot act on is worth nothing. At $B = 14$ it gains 0.042 over the best fixed level by the retry path $0.1 \times 0.6 \times 0.7$, the bound of Theorem 5 attained exactly, while the ceilings 0.854 and 0.957 match Corollary 1 to every digit: the cheap model never attains 90% and the mid model never attains 98% at any budget, which answers "can we just give it more tokens?".

It is easy to overclaim about adaptivity, so we do not: at the obvious target 0.90 there is *no* capacity gap, since $\pi^\star$ and the fixed mid level both reach it at $B = 11$, and the strict gap appears only at 0.92. Appendix L gives the full sweeps and two further findings: budget and deadline are complements, so tokens are nearly worthless when time binds; and consumption variance is *not* monotone, harmless while the distribution stays inside the feasible region and catastrophic once its upper tail crosses the boundary at which the budget can no longer fund the action, which §6 measures on real logs.

6 The bill, on 8,190 published runs

Three of this paper's claims are claims about an *ordering*, and an ordering is the easiest thing in the world to manufacture with a stylized three-point menu, so they are tested on independently

developed agents, on independently written tasks, externally scored by the benchmark's own grader. We executed nothing and reran nothing: we analyzed AppWorld's published logs (Trivedi et al., 2024), *reconstructed* each run's token cost from the code the agent wrote, the output the environment returned and the prompt-assembly code in the same repository, and *replayed* those costs against counterfactual budgets, so the curves measure fixed policies under a budget, not budget-aware ones.

The corpus, and the allowance already in it. AppWorld publishes the raw output of its own leaderboard entries: four agent architectures across up to four models each, 14 configurations, with *every configuration run on the same tasks*: 168 in `test_normal` and 417 in `test_challenge`, so 8,190 trajectories and 212,550 turns (Appendix M). They do not record tokens; they record something better for the present purpose, the allowance itself. AppWorld's configuration files declare, per architecture, a cap on model calls, a cap in dollars, a cap on steps and two caps in characters (`max_llm_calls: 100`, `max_llm_cost: 10.00`, `max_steps: 15`): four architectures, four currencies, and not one of those numbers reported beside the score. The allowance *is* written down, in a file nobody reads beside the leaderboard, and it binds: in three of the four architectures the declared counter is never reached while run lengths concentrate at a lower, undocumented ceiling, ReAct declaring 100 model calls and stopping at 71 turns on 61.3% of one configuration's tasks.

Reconstructing the bill, and checking the reconstruction (E6). These agents are deterministic text pipelines whose prompt-assembly code ships in the same repository, so the charged input at turn k is the instruction header plus every earlier code block and tool result, assembled and trimmed by the harness's own routine. Over n turns that is $n \cdot \text{header} + \sum_k (n-k)s_k$, with s_k the size of turn k's code and tool output: quadratic in the turn count, dominated by whatever the tools printed, and totalling 1.37B charged input tokens against 16.8M tokens of code the agents wrote. A reconstruction needs ground truth, and OpenManus (Liang et al., 2025) is the only corpus here that reports provider-side usage beside every model call; on its 30 GAIA runs and 940 calls the shape is confirmed and two numbers carry forward. Charged input grows by 619 tokens per call, with a positive slope in 30 of 30 runs, and the cumulative bill is better explained by the square of the call count than by the count (R^2 0.879 against 0.795). And **completion tokens are 1.40% of the bill**: the reasoning trace, the thing a thinking-budget parameter buys, is a rounding error next to tool results re-entering the context. That also prices the paper's own abstraction, since a constant c_t is wrong by a factor of 13.2 within one run (median ratio of last call to second, 19,949 against 1,496 tokens). The direction matters more than the size: a constant cost understates late steps, so it understates exhaustion, and the model is optimistic exactly where the risk is.

The capacity curve, measured, and the ordering it destroys (E7, E8). For configuration a and allowance B, let $\sigma_a(B)$ be the share of tasks it both solved and could pay for out of B. These are *pathwise replay curves under reconstructed costs*, and the qualifier is load-bearing: they are exact given the logged trajectory and the reconstruction rule, because none of these agents can see a budget, so an allowance can do exactly one thing to a logged run, cutting it off where the bill first exceeds B; but they are *not* an estimate of what these configurations would do if deployed under a cap.

Figure 2 is the result (Table M6 in Appendix M gives all fourteen configurations), and the right-hand end of every curve is that configuration's published score. **The published leaderboard ranks configurations only by their unconstrained, high-budget success rates.** ReAct/GPT-4o tops it at 0.488 and has zero observed success below 23,126 tokens, its onset β in the sense of Theorem 3, while FullCode/GPT-4o, ranked third at 0.339, delivers its first success at 4,320 and all of them by 10k. At $B = 10$k the leaderboard's first, second, fourth, fifth, sixth and seventh entries all return zero while its third returns 0.339; and because every configuration ran the same 168 tasks that comparison is paired and exactly testable, the discordant counts being 0 tasks solved-in-budget by ReAct/GPT-4o but not by FullCode/GPT-4o against 57 the other way, so an exact McNemar test gives $p = 1.4 \times 10^{-17}$ and the pointwise 95% Clopper-Pearson intervals $[0.000, 0.022]$ and $[0.268, 0.416]$ are disjoint. Proposition 1 gives ± 0.105 uniformly over a whole curve, wider than most vertical gaps, so it is the paired structure and not the marginal band that makes the crossings significant. Across the fourteen configurations 41 ordered pairs cross, behaving as Theorem 3 requires: the onset and ceiling orderings disagree, because the architectures that score best spend most before delivering anything. The theorem was proved on a three-point menu; 41 instances of it sit on somebody else's leaderboard.

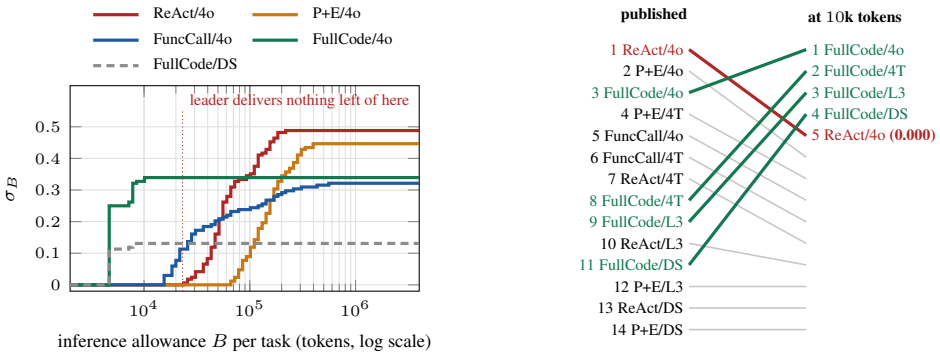

Figure 2: **The leaderboard is not the ordering. (a)** Each curve is one AppWorld entry's σ_B: the share of the same 168 tasks it both solved and could pay for out of B tokens. The right-hand end is its published score, and the published ordering is the ordering *only* there. **(b)** The same fourteen configurations ranked by published score (left) and by what they deliver at $B = 10$k (right). Ten of the fourteen, including the published first and second, return *exactly* 0.000 at that allowance; the four whole-program agents, published third, eighth, ninth and eleventh, take the first four places. Architectures are abbreviated (P+E, FuncCall, FullCode) and models by family (4o, 4T, L3, DS); ties among the zeros are broken by ceiling. The reordering is not noise: at $B = 10$k the paired discordant counts between the published first and third are 0 against 57, an exact McNemar $p = 1.4 \times 10^{-17}$.

Ceilings, and what choosing on capacity is worth (E9). The flat right-hand tails are *not* evidence for Theorem 2: on fixed logged trajectories the right-hand limit *is* the published score by construction, so the flatness is an identity rather than a measurement. What the data supplies is the ceilings' values and their consequence: on `test_challenge` *no* configuration reaches 0.40 at any budget. Inverting the curves (Table M7, Appendix M) prices choosing on capacity rather than accuracy: to complete 30% of `test_normal` the configuration a practitioner would pick from the score needs 64k tokens per task and the cheapest one reaching the same target needs 7k, an 88% **saving for the same delivered reliability** (91% at the 20% target). The leaderboard pick is cheapest only at the 0.40 target, which only two of the fourteen reach: choosing by score is right exactly when almost nothing meets the target, not the regime a token cap puts you in.

Half of a benchmark's failures are not wrong answers (E10, E11). The metric that motivates the third absorbing state can be computed here because these runs record one bit the score throws away. An AppWorld task ends when the agent calls `supervisor.complete_task`: one that called it and was marked wrong concluded and was wrong, one that never called it was still working when its harness stopped it, and the scorer records both as `success: false`. The bit is a proxy with one check available, that a scored success must have concluded, and it passes at 1.000 on all 8,190 trajectories. That establishes sensitivity, not specificity, since not concluding is also consistent with a step cap, a tool failure or a stalled policy (Cemri et al., 2025); these runs are therefore *non-concluded*, not *exhausted*. Of 6,913 recorded failures, 3,752 are errors and 3,161**, or** 45.7%**, are non-conclusions**: nearly half of what this benchmark reports as an agent getting the answer wrong is an agent that never gave one. Exactly 1,300 of those (41.1%) stopped at their configuration's observed ceiling and were demonstrably counted out; for the other 1,861 the logs are silent. The per-configuration spread, 0.068 to 0.939, is the useful part: on ReAct/GPT-4o all 22 non-conclusions sit at the ceiling and on Plan-and-Execute with DeepSeek 1 of 155. So it is a property of the configuration, not the benchmark, and it decides which repair to buy. E11 locates the allowance above which errors outnumber non-conclusions: 7k tokens for one configuration, 594k for another, none at all for two of the three DeepSeek ones (Appendix M).

Which escalation trigger actually pays (E12). Because every configuration ran every task, a two-stage policy over them is evaluated exactly rather than sampled. Deployed cascades consult one trigger and ignore the other (Chen et al., 2023; Ong et al., 2024): *confidence* escalates when the first stage failed to declare itself finished, *budget* whenever what is left still covers the second stage. The confidence trigger is worth 0.000 against the budget trigger's 0.077: the signal nobody consults is the one with predictive value, because the cheap architecture declares itself finished on 90.5% of

tasks and is wrong on 71.7% of those, so a trigger keyed to it fires on the remaining 9.5% and sits out almost every case where escalating would have helped. The pair is the best of all 182, and an oracle escalating on the first stage's actual correctness beats the budget rule by 0.060 (Table M9).

Where the model is wrong, and what the benchmark does not settle (E13, E14). Both of the model's simplifications are tested here, and the one that matters errs safely: its constant-q parameterization is refused by all thirteen configurations with enough of both outcomes to ask, because cheap runs are the ones that worked, so a constant q makes the model *understate* the measured curve by up to 0.096 and $B^\star$ is an upper bound. Recomputing every bill with the history cap removed makes the untrimmed corpus $7.3\times$ the one charged, the *median* barely moving while the 99th percentile on the worst arm goes from 913k to 115M tokens: E5 measured, and the reason to cap what a tool may print. Four caveats bound all of the above: the tokens are *reconstructed, not billed*, so every comparison is within one tokenizer and in tokens rather than dollars; *one architecture is excluded* from the per-call statistics, its log flattening multiple tool calls into one message; *the replay imposes budgets on policies that were not budget-aware*, which makes $\sigma_a(B)$ exact and stops it measuring *adaptive* ones; and *the escalation result is about this corpus's triggers*.

What a benchmark should report. Making the above routine needs one small change. A leaderboard that publishes a single score publishes the *ceiling* $\bar{\sigma}$, which by Theorem 3 is one endpoint of an ordering the reader cannot reconstruct. We propose publishing the **capacity triple** $(\beta, \bar{\sigma}, \eta)$ beside it: the *onset* β, the least allowance at which the configuration delivers anything; the *ceiling* $\bar{\sigma}$ already reported; and the *exhaustion share* η, the fraction of failures that produced no answer. All three come from artifacts harnesses already write, β and $\bar{\sigma}$ from per-trajectory cost and η from the termination bit, and together they turn Theorem 3 into a test a reader can run without rerunning anything: whenever a lower-scoring configuration has the earlier onset, the two curves must cross. The test is sufficient, not necessary, since curves with equal onsets can still cross by climbing at different rates.

7 Related work and limitations

Agentic evaluation is where this paper is aimed. The benchmarks in widest use score whether a trajectory ended correctly (Mialon et al., 2024; Jimenez et al., 2024; Liu et al., 2024; Yao et al., 2025) and none publishes a per-trajectory cost beside it, so a run correct only with twice the allowance scores identically to one that was affordable. The closest prior critique is Kapoor et al. (2024), which argues that agent evaluations conflate accuracy with cost and proposes a cost-accuracy Pareto frontier. We share the diagnosis and differ in the object: a frontier plots where each configuration *landed*, while a capacity curve is a function of a stated cap and says what each system would deliver *under* it. The difference is Theorem 3: a frontier cannot exhibit the onset β, and the onset decides whether the ordering survives.

Cost-aware routing, cascades and adaptive inference-time computation are the closest operational relatives: cascades escalate on a learned confidence signal (Chen et al., 2023; Yue et al., 2024), routers select a tier per query (Ong et al., 2024), adaptive sampling stops early on agreement (Aggarwal et al., 2023), and reasoning-effort settings, test-time search (Snell et al., 2024; Wang et al., 2023) and input-adaptive allocation (Damani et al., 2025) make spend-versus-quality an explicit control, and Stroebl et al. (2024) show that resampling against an imperfect verifier saturates, which is Corollary 1 at one step. All of it optimizes *one* query or step; the missing piece is the mission-level accounting in which a step-local optimum can be mission-fatal. *On the formal side* none of the machinery is new: metareasoning (Horvitz, 1988; Russell and Wefald, 1991; Zilberstein, 1996), constrained MDPs (Altman, 1999), stochastic shortest paths (Bertsekas and Tsitsiklis, 1991) and reward-augmented stochastic Petri nets (Ajmone Marsan et al., 1995; Ciardo et al., 1993) supply every piece. The contribution is the identification of the agentic quantity they should be pointed at, the outcome partition that makes exhaustion nameable, the two orderings of Theorem 3, and the demonstration that the resulting orderings differ from evaluation's.

Limitations. The budget is scalar, a call's cost is assumed known in advance, and renewable resources are out of scope (Appendix G); the measured curves also impose a budget on agents that could not respond to one.

Reproducibility statement

The model analyzed in §5 is a deterministic computation specified by Definition 1, the parameters in §2 and the uniformization tolerance in Appendix L; no random seed enters it. Assumptions and full proofs of all results are in Appendix J. The accounting model, the tokenizers used per model family, the validation corpus and the filters behind every count in §6 are given in Appendix M. The supplementary material contains `model_exact.py`, which recomputes every number in §5 and Tables L3 and L4 from the parameters alone, together with its output; and `stats_bench.py`, which recomputes the interval estimates, the uniform band and the exact paired test of §3 and §6 from the per-configuration shares printed in Table M6. The reconstruction pipeline itself is not redistributed, because it reads AppWorld's published logs and prompt-assembly code rather than ours; Appendix M specifies it in full.

Ethics statement

This work analyzes publicly released benchmark artifacts and involves no human subjects, no personal data and no new model. One risk is worth naming: a deployment tuned to the edge of its capacity curve fails by silent exhaustion rather than by error, a mode current evaluations do not distinguish (§6), so we recommend reporting the exhaustion share beside the success rate.

References

P. Aggarwal, A. Madaan, Y. Yang, and Mausam. Let's sample step by step: adaptive-consistency for efficient reasoning and coding with LLMs. In *EMNLP*, 2023.

M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

M. Cemri, M. Z. Pan, S. Yang, L. A. 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.

L. Chen, M. Zaharia, and J. Zou. FrugalGPT: how to use large language models while reducing cost and improving performance. arXiv:2305.05176, 2023.

G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi. Automated generation and analysis of Markov reward models using stochastic reward nets. In *Linear Algebra, Markov Chains, and Queueing Models*, pages 145–191. Springer, 1993.

M. Damani, I. Shenfeld, A. Peng, A. Bobu, and J. Andreas. Learning how hard to think: input-adaptive allocation of LM computation. In *ICLR*, 2025.

E. J. Horvitz. Reasoning under varying and uncertain resource constraints. In *AAAI*, pages 111–116, 1988.

K. Jensen and L. M. Kristensen. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Springer, 2009.

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.

S. Kapoor, B. Stroebl, Z. S. Siegel, N. Nadgir, and A. Narayanan. AI agents that matter. arXiv:2407.01502, 2024.

X. Liang, J. Xiang, Z. Yu, J. Zhang, S. Hong, S. Fan, and X. Tang. OpenManus: an open-source framework for building general AI agents. https://github.com/FoundationAgents/OpenManus, 2025.

X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. AgentBench: evaluating LLMs as agents. In *ICLR*, 2024.

P. Massart. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *Annals of Probability*, 18(3):1269–1283, 1990.

G. Mialon, C. Fourrier, C. Swift, T. Wolf, Y. LeCun, and T. Scialom. GAIA: a benchmark for general AI assistants. In *ICLR*, 2024.

M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Trans. Computers*, 31(9):913–917, 1982.

T. Murata. Petri nets: properties, analysis and applications. *Proc. IEEE*, 77(4):541–580, 1989.

I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. RouteLLM: learning to route LLMs with preference data. arXiv:2406.18665, 2024.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

S. Russell and E. Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1–3):361–395, 1991.

C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. arXiv:2408.03314, 2024.

B. Stroebl, S. Kapoor, and A. Narayanan. Inference scaling fLaws: the limits of LLM resampling with imperfect verifiers. arXiv:2411.17501, 2024.

H. Trivedi, T. Khot, M. Hartmann, R. Manku, V. Dong, E. Li, S. Gupta, A. Sabharwal, and N. Balasubramanian. AppWorld: a controllable world of apps and people for benchmarking interactive coding agents. In *Proc. ACL*, 2024. Best Resource Paper.

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.

S. Yao, N. Shinn, P. Razavi, and K. Narasimhan. τ-bench: a benchmark for tool-agent-user interaction in real-world domains. In *ICLR*, 2025.

M. Yue, J. Zhao, M. Zhang, L. Du, and Z. Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *ICLR*, 2024.

S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

A THE USE OF LARGE LANGUAGE MODELS

LLMs were used only as a general-purpose writing and editing aid on this manuscript (phrasing and LaTeX formatting). They played no role in research ideation, in producing or analyzing any of the data, or in writing the analysis harness, and they are not a component of the method. The authors take full responsibility for all content.

HOW TO READ THIS APPENDIX

None of the three readers it serves needs all of it. A reader with no background should start at Appendix D, which assumes nothing: places, marks, the one firing rule, what putting a budget in the state means, what an absorbing Markov chain is, why backward induction works here when it usually needs iteration, and the headline example worked by hand. A reader who wants to know whether this applies to *their* agent should read Appendix H, which turns an ordinary agent configuration into a token-aware net line by line and says which lines become costs, and then Appendix I, which works four deployed systems and states what the optimal policy reduces to in each one's own vocabulary. A reader who wants to know why this formalism should read Appendix G, which argues it against writing the decision process directly, against constrained Markov decision processes, against queueing and cost-calculator treatments, against cost-aware routing, and against simply measuring, and then says what the net does *not* give; Appendix E expands the related-work comparisons the body compresses, and Appendix F is the decision procedure a practitioner would follow. A reader checking the claims should read Appendix J for the proofs, Appendix L for the protocol behind the exact experiments of §5, and Appendix M for the corpus, the accounting model, and the validation behind §6. Appendix K treats within-call consumption, Appendix N is the glossary and symbol table, and Appendix A is the statement on the use of large language models.

B NOTATION AND ASSUMPTIONS, RESTATED

The appendices are read out of order, so everything they invoke is collected here. Nothing in this section is new.

The object. A *budget-augmented stochastic Petri net* is $\mathfrak{B} = (P, T, Pre, Post, \Lambda, c, M_0, B)$: places, transitions, the arc matrices with incidence $C = Post - Pre$, a rate function $\lambda_t(M, b) > 0$ per transition, a token cost $c_t \in \mathbb{N}$ per transition, an initial marking and an initial budget (Definition 1). There are no colors and no age annotations; the class is stated narrowly on purpose, so that Theorem 4 is a claim about every model in it. Petri-net texts say *token* for a mark; here *token* always means the language-model unit, and a dot in a place is a **mark**.

The state, and the gate. The state is the pair (M, b) with $b \in \{0, \dots, B\}$ the remaining budget. The enabled set is $T_{\mathrm{En}}(M, b) = \{t : M \geq Pre(:, t) \text{ and } b \geq c_t\}$ and the generator it induces are both Eq. (1). The second conjunct is the entire construction: an unaffordable action is not penalized, it is *absent*, so the constraint holds under every assignment of rates rather than in expectation.

Which resource this is about. A cumulative, non-renewable, per-mission inference budget, measured in tokens or the money they cost. Not a context window, which constrains occupancy and can be freed; not a rate limit, which replenishes; not a per-call output cap, which binds inside one call. Those reduce to this one only under the conditions §2 states, and every proof here depends on non-renewability.

Three outcomes. Σ is mission success, F is mission error, and $\mathcal{E}$, defined in §2, is *token exhaustion*: the states where the workflow still has moves and the budget can pay for none of them. Ordinary deadlocks, where the net has nowhere to go at any budget, are classified with F. The operational content of the split is that F calls for a better model and $\mathcal{E}$ for a bigger allowance or a cheaper plan, and a success rate alone does not distinguish them.

Race versus choice. Two processes live on the same net. Under a fixed configuration the competing transitions are an ordinary stochastic race and (1) decides which fires. Under a *policy* a router picks

Table C1: The derivation. Column one is agentic, column two is the modeling step, column three is a standard quantity on a Markov chain. Nothing in column three is new mathematics; the contribution is that writing an agentic architecture this way makes those familiar quantities *mean* the things in column one.

Agentic phenomenon	**In the net**	**Chain quantity**	**Metric**
an LLM call consumes tokens	a transition with cost c_t drawing from p_{budget}	the state becomes (M, b)	n/a
a token cap, context window, or spend ceiling	the initial marking B of the budget place	a parameter of the chain	all
choosing a model tier, or a reasoning-effort setting	competing transitions from one place with different (c, λ, q)	the action set $A(M, b)$ under a policy; a race under a fixed configuration	σ_B
the agent runs out mid-task	the absorbing set $\mathcal{E}$	absorption mass in $\mathcal{E}$	η
retries after a failed check	a workflow cycle that still decreases b	expected visits before absorption	$\bar{C}$
"how much quota does this workload need?"	least B whose absorption in Σ meets the target	inverse of the absorption curve	$B^\star$
variable-length generation	branch splitting with weights $p_i(k)$	a mixture over transitions	$\sigma_B, \bar{C}$
a routing policy: which model for which step	the action set at a place	an optimal policy on the induced DAG	$\pi^\star$

the action set $a = \pi(M, b) \in A(M, b)$ first, and only then does anything stochastic happen; the object is then a continuous-time Markov decision process. "Always H" means the constant policy, $\pi^\star$ means the maximizer.

The metrics. Token-constrained success $\sigma_B(D)$; exhaustion probability $\eta(B, D)$; expected consumption $\bar{C}(B, D)$; and the *capacity curve* $B^\star(D, \varepsilon) = \min\{B : \sigma_B(D) \geq 1 - \varepsilon\}$ (§3). All four are absorption or transient quantities, computed by a linear solve or by uniformization, with B a parameter to sweep.

The results. Theorem 1 (inversion interval): with κ the cheapest completion tail after the reasoning step, a strictly more accurate level returns mission success exactly 0 while a cheaper one is positive, for every remaining budget in $[c_\ell + \kappa,\ c_{\ell'} + \kappa)$, an interval of width exactly the price premium, so no function of per-step accuracies orders mission outcomes. Theorem 2: under (C1) to (C3) each fixed configuration has a reliability ceiling no budget crosses, with the closed form of Corollary 1 for a verify-retry loop, so buying tokens and buying reasoning quality are not substitutes. Theorem 3: two capacity curves must cross whenever the higher-ceiling configuration has the later onset, so agreement of the ceiling and onset orderings is necessary for a score ranking to hold at every budget. Theorem 4: if every cycle in the net contains a cost-bearing transition then the augmented process is acyclic in b, so the optimal policy is exact by one backward pass with no convergence criterion. Theorem 5: that policy dominates every fixed level at every budget, by a margin bounded in closed form. Proposition 1: the measured capacity curve is an empirical sub-distribution function, so one Dvoretzky-Kiefer-Wolfowitz band covers it uniformly and inverts to a confidence interval on $B^\star$.

The experiments. E1 to E5 are exact analyses of the model: absorption by backward induction, deadline quantities by uniformization, no sampling. E6 onward read a public corpus of published agent runs and are measurements, with the accounting model and its limits in Appendix M. An exact 0 in E1 to E5 is unreachability; a small number later is a frequency.

C From agentic phenomena to budget metrics

This appendix is the argument that the formalism earns its place. Each row of Table C1 begins with something a builder of agentic systems would recognize, states what that thing *is* once the architecture is written as a token-aware net, and reads off the quantity that becomes computable as a result. The metrics of §3 are the last column.

Three properties of this derivation are what a practitioner gets from it.

The design variables are explicit. The budget B and the per-step costs c_t are not properties of any model; they are numbers an architect sets and a provider bills for. Once they are in the state they can be swept, and §5 sweeps them. An evaluation that reports only task accuracy has nowhere to put B at all, which is why "how much quota does this agent need" is currently answered by watching the bill.

Structural and stochastic claims stay separate. That an unaffordable action cannot fire is a reachability statement and holds for every assignment of rates and every task distribution. That the mission succeeds with probability 0.900 is an estimate that depends on measured correctness and latency. A builder should know which of the two a given guarantee is.

The metrics compose the way the system does. All of them live on one chain, so they interact in the model exactly as they interact in production: a retry budget consumes tokens that a downstream tool call needed, a deadline makes the slow-but-accurate tier less attractive, and adding a critic pass both raises correctness and spends the allowance twice. Benchmarking each concern separately cannot see any of that.

Which systems this covers. The running mission is an ops agent, but the shape is generic. What changes across architectures is only what the budget and the competing spend mean.

Architecture	**The allowance B**	**What competes for it**	**The deadline**
RAG agent	per-request token cap	retrieve more passages vs. answer now	user latency budget
coding agent	session context or token budget	explore more files vs. write the patch	CI or review window
deep research	spend ceiling per report	read another source vs. synthesize	report deadline
multi-agent crew	a shared organizational quota	which agent is funded to think hard	delivery SLA
ops or incident agent (§2)	control-plane rate quota	deeper diagnosis vs. mitigate now	incident SLA

The fourth row generalizes least comfortably and is worth naming. In a crew the allowance is shared across agents rather than across steps of one agent, so the competition is between *principals*, and the question of who gets to think hard is an allocation problem with an owner. The model handles it, since a shared budget place is a shared budget place, but the policy it produces is then a statement about organizational priority rather than about a single agent's control loop.

Appendix I works four of these rows out. Three points from it are worth previewing. In a coding agent the dominant spend is tool *results* rather than reasoning, so the binding trade-off is exploration against action and the heavy upper tail of p_i is set by how much a test run prints, which no reasoning-effort setting controls. In deep research the terminal action costs more the more the earlier steps gathered, so the number of sources is properly an *output* of the budget rather than a prompt parameter. And in a support agent the escalation decision, today triggered by a confidence signal alone, is a threshold on remaining budget as well: the two signals are independent and one of them is unused.

D BUDGETS, NETS, AND MARKOV CHAINS FROM SCRATCH

This appendix assumes no background. It says everything in words before symbols, and a reader who knows Petri nets and absorbing Markov chains can skip it.

The picture. A *net* has two kinds of nodes. A *place*, drawn as a circle, holds some number of *marks*, drawn as dots. A *transition*, drawn as a bar, is an event. Arcs run between them, never between two places. The list of how many marks sit in each place is the *marking*, and it is the entire state: nothing is stored anywhere else.

Read a place as a condition that holds or a resource that can be counted, and a transition as something that happens. In Figure 1, a mark in p_{ctx} means the agent has retrieved its context, and the transition t_{M} is the event "reason using the mid-tier model".

The one rule. A transition may fire when every place with an arc into it holds enough marks. Firing *removes* those marks and *adds* marks to its output places. Two consequences do all the work here.

A transition needs *all* its inputs, so t_{act} cannot fire without both a verified conclusion and enough budget; this is conjunction, made structural. And firing *spends* the inputs: after t_{H} takes ten marks from the budget place, those marks are gone, not marked used. That is what makes a net able to express running out of something.

What "putting the budget in the state" means. An ordinary model of this workflow would track which step the agent has reached. We track a pair: the step, *and* how much budget is left. Think of it as a bank balance carried alongside the itinerary. The reason to carry it is that the same step can be either fine or fatal depending on the balance, and a model that does not carry the balance cannot tell those apart.

The gate is the important part. We do not charge a penalty for overspending; we make the overspending transition *not fire at all*. An action the balance cannot cover is not an action that is attempted and refused, in the same way that an API call you cannot pay for is not a call that fails, it is a call that is never made.

Rates, and what a Markov chain buys. Real steps take time, and how long is uncertain. We give each transition a *rate*: a transition with rate λ takes on average $1/\lambda$ time units, and when several transitions are enabled they race, the fastest one firing. A process built this way is a *continuous-time Markov chain*, which is simply a system that jumps between states at random times with the property that where it goes next depends only on where it is now, not on how it got there.

Three classes of states here are *absorbing*: once reached, the process stays. They are mission success, mission error, and budget exhaustion. The quantity we want, "what is the chance this mission succeeds", is then the probability of being absorbed in the first of the three, which is found by solving one system of linear equations. No simulation is involved, and the answer is exact.

Deadlines, and the one piece of machinery with an opaque name. A deadline turns "will it succeed" into "will it succeed *by* D", which is a question about the distribution of states at a finite time rather than about eventual absorption. That distribution is $\pi(0)e^{\mathbf{Q}D}$, where $\mathbf{Q}$ is the matrix of rates, and it is computed here by *uniformization*, which deserves one sentence because the word explains nothing. Pick a Γ at least as large as the fastest total rate anywhere in the chain and write $P = I + \mathbf{Q}/\Gamma$, which is then an ordinary probability matrix; running P for a Poisson(ΓD) number of steps gives exactly the same distribution as letting the original process run for time D. Since the Poisson weights are non-negative and sum to one, stopping the sum once the remaining weight is below 10^{-12} leaves an error *bounded* by 10^{-12} rather than estimated at it. That is the sense in which the deadline numbers here are exact.

Three outcomes rather than two. Separating exhaustion from error is the practical point of the whole construction. A benchmark that reports only success and failure conflates "the model reasoned badly" with "the agent ran out of allowance". The first calls for a better model and the second for a larger budget or a cheaper plan, and no amount of the wrong remedy fixes the other problem.

Working backwards. When the agent has a genuine choice, here which reasoning level to use, we want the best rule. Ordinarily finding the best rule in such a system requires iterating a computation until it stops changing, because choices can loop back on themselves.

That cannot happen here, and the reason is visible in Figure D1: every model call spends budget and none returns any, so the balance only ever falls. A state can therefore never recur, and the states can be lined up in order of remaining balance. We then start at the poorest states, where the answer is obvious, and work backwards: knowing the value of every state with a smaller balance, the best choice at a richer one is whichever action leads to the best mix of them. One pass, exact, no convergence test.

Working the headline example by hand. Budget 14. Observing costs 2, leaving 12. Now choose.

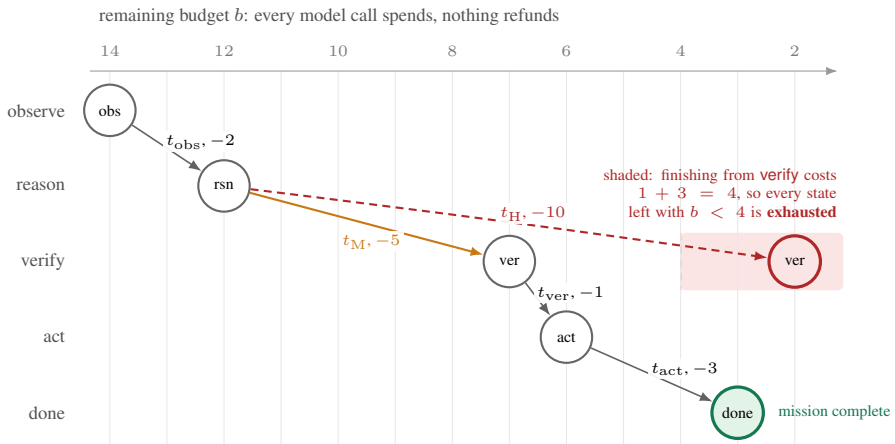

Figure D1: Putting the budget in the state, drawn on the $B = 14$ instance. A state is a pair: which stage the mission has reached (rows) and how much budget is left (columns), so a state's horizontal position *is* its remaining budget. Both arrows out of rsn are legal and the upper one is the more accurate reasoner. The mid level costs 5 and lands at $b = 7$, with enough left to verify and act; the deep level costs 10 and lands at $b = 2$, from which finishing costs 4 and nothing affordable remains, so that branch is stranded with probability one. This is Theorem 1 as a picture: the reasoner that is right more often is the reasoner that cannot pay for what it concluded.

Take the deep level: it costs 10, leaving 2. Verification costs 1 and acting costs 3, so 4 are needed and 2 remain. No transition can fire. The mission is stranded, and this is certain, not merely likely, so the success probability is exactly 0.

Take the mid level: it costs 5, leaving 7. Verification leaves 6, acting leaves 3, and the mission completes whenever the reasoning was right, which is 90% of the time. Rounding aside, that is the 0.900 in Table L3.

The deep level is the better reasoner by every measure an evaluation reports. It is also, at this budget, the only choice that guarantees failure. Nothing subtle is happening: it simply spends money it needed later.

E Extended related work

The body states the positioning; this appendix gives it in full.

Cost-aware routing, cascades and adaptive inference-time computation are the closest operational relatives: cascades escalate on a learned confidence signal (Chen et al., 2023; Yue et al., 2024), routers select a tier per query (Ong et al., 2024), adaptive sampling stops early when agreement is high (Aggarwal et al., 2023), and reasoning-effort settings, test-time search (Snell et al., 2024; Wang et al., 2023) and input-adaptive allocation (Damani et al., 2025) make spend-versus-quality an explicit control, with Stroebl et al. (2024) showing that resampling against an imperfect verifier saturates, which is Corollary 1 at the level of one step. All of it optimizes *one* query or step and takes the escalation trigger as the design variable; our E12 is a direct measurement in that frame, and its finding is about which variable to condition on, not a new router. What is missing is the mission-level accounting in which a step-local optimum can be mission-fatal.

On the formal side none of the machinery is new, and the paper is better for saying so: metareasoning frames the problem of deciding how much to deliberate (Horvitz, 1988; Russell and Wefald, 1991; Zilberstein, 1996), while constrained Markov decision processes (Altman, 1999), stochastic shortest-path problems (Bertsekas and Tsitsiklis, 1991) and reward-augmented stochastic Petri nets (Ajmone

Marsan et al., 1995; Ciardo et al., 1993) supply the pieces, the acyclicity of a depleting resource being folklore there. The contribution is the identification of the agentic quantity that machinery should be pointed at, the three-way outcome partition that makes exhaustion nameable, the two orderings of Theorem 3, and the demonstration that the resulting orderings differ from the ones evaluation reports (Appendices G and F).

F USING THIS ON A SYSTEM YOU OWN

The metrics order the decisions, and the order matters because the cheap diagnostic comes first.

Compute η first. If failures are non-conclusions rather than wrong answers, a model upgrade is not obviously the repair and the question is capacity or plan cost. The caveat runs one way: a stronger model may also conclude in fewer calls, so it can reduce non-conclusions too; what η rules out is the *assumption* that the failures were reasoning failures. The computation is cheap, and §6 is the recipe: any harness that records whether the agent declared itself finished already separates the two, and on the one benchmark where we could check it the bit was right on every scored success. The crossover budget of E11 then says which purchase is the live one at the allowance actually in force.

Then compute the capacity curve $B^\star$ for each candidate configuration. If a configuration's ceiling is below target, discard it regardless of price: no allowance will move it (Theorem 2). Among the survivors, compare $B^\star$ rather than accuracy, since that is the quantity denominated in money, and record the onset β as well as the ceiling, since by Theorem 3 it is the onset that decides whether a ranking by score survives the budget you actually have.

Finally, if the budget is near $B^\star$, adopt the adaptive policy. It is a lookup table over remaining budget, computed by one backward pass (Theorem 4), and it is worth most exactly where the fixed configurations are forced to choose between affordability and accuracy. Where one configuration is unambiguously right it ties, which is the correct behaviour: adaptivity is insurance against the budget moving, not free performance.

G WHY A TOKEN-AWARE NET, AND NOT SOMETHING ELSE

The construction of §2 is deliberately small, which invites the question of whether it was needed at all. This appendix takes the alternatives one at a time and says what each loses on the running mission. The honest summary is that one of them, writing the decision process directly, loses nothing mathematically and a good deal practically, and that the rest lose something mathematical.

Write the Markov decision process directly. This is the strongest alternative and it deserves the first answer: everything proved here *is* a statement about a finite MDP, and a reader who prefers to work with $(\mathcal{S}, A, P, r)$ may. What the net supplies is not extra power but three practical things. The state space is *generated* rather than enumerated: $|P|$ places and $|T|$ transitions describe a reachable set that a hand-written MDP would have to list, and the gap is exponential as soon as the workflow has concurrent branches, since k independent branches give 2^k combinations of local states and $2k$ net elements. The hypothesis of Theorem 4 is *structural*: "every cycle contains a transition with $c_t > 0$" is a property of the incidence matrix, checkable before any state is built, whereas the corresponding MDP condition is a statement about a graph one must construct first. And the budget constraint is an *enabling rule* rather than a side condition, so it is inherited automatically by reachability, by the policy computation and by every sweep, instead of being re-imposed in each. A net here is a modelling language for a class of MDPs; the claim is convenience and checkability, not expressive power, and it is worth saying so plainly.

A constrained MDP with an expected-cost constraint. This is the standard formulation and it is the wrong one, which is worth being precise about because the mistake is easy. A constrained MDP maximizes reward subject to $\mathbb{E}[\text{cost}] \le B$, usually through a Lagrangian. That constraint is on the *average* trajectory, so it permits individual trajectories to overspend, and overspending is not a thing that happens in the systems being modelled: a provider does not let a mission run 20% over its cap on some runs to compensate for frugal ones. Worse, the failure this paper is about, that the mission is stranded and that this is a distinct outcome from being wrong, is invisible in a

formulation where cost is a soft penalty, because a penalized policy still finishes. Putting b in the state is the per-trajectory hard-constraint formulation, and the price is a state space multiplied by B; the acyclicity of Theorem 4 is what makes that price affordable, so the two decisions are not independent.

A queueing or performance model. Queueing networks are the natural home for latency and throughput under service rates, and they are a poor home for this, because the budget is not a server. Nothing queues for it, it grants no service time, and above all it does not replenish. The product-form and steady-state results that make queueing theory powerful assume some form of renewal, and a non-renewable pool has no steady state to speak of: every trajectory ends. Where a resource *does* replenish, as a rate limit over a long horizon does, queueing is the right tool and this paper is not, which §7 states as an open direction rather than a claim.

A cost calculator. Multiply expected calls by unit price and compare against the ceiling. This is what most teams do, it is not foolish, and it computes $\bar{C}$, the mean. The paper's sharpest empirical finding is precisely that the mean is the wrong statistic: E5 holds the mean consumption fixed and varies only its spread, and success falls from 0.980 to 0.496 once the upper tail crosses the feasibility boundary. A calculator reports that the mission is comfortably funded in both cases, because in expectation it is. Feasibility is not a property of an average, and no arithmetic on means recovers it.

Cost-aware routing and model cascades. A well-developed line of work routes each query to the cheapest model that will answer it, escalating on low confidence. It is complementary rather than rival, and the difference is exactly the coupling this paper is about: a router optimizes *per query*, treating each as independent, and a cascade never strands itself, because there is no later step whose affordability the current choice must protect. The running mission's whole content is that spending on reasoning removes budget the *action* needed, which is a statement about a sequence and cannot be expressed one query at a time. Where a routing policy is embedded in a workflow, it becomes the action set $A(M, b)$ of §2 and $\pi^\star$ is the budget-aware version of it.

Anytime algorithms and metareasoning. The closest intellectual relative: classical metareasoning trades deliberation against decision quality under a time-dependent utility. Two differences matter. The resource there is time and the penalty is soft, a discount on utility, whereas here the constraint is a hard gate and running out is a third absorbing outcome rather than a low score. And metareasoning is usually about one decision, whereas the object here is a workflow whose steps compete for one pool, which is where the non-monotonicity of Theorem 1 comes from. The shared instinct, that more computation is not always better, is the same, and this paper's contribution is to make it a computation rather than a caution.

Just measure it. Run the configuration and see. For one configuration one should, and §6 does exactly that on a public corpus. The questions a builder faces are counterfactual and combinatorial: what budget does this mission need for 90% reliability, which step deserves the marginal token, is the cheap model adequate here, does a retry budget buy more than a better reasoner. Answering those by deployment means paying for the failures and running the sweep, and the sweep is exponential in the number of choices. It is also confounded: a measured success rate does not separate F from $\mathcal{E}$ unless the harness was built to record which one it was, and most were not.

Context management as a rival. Compaction, summarization at a watermark, and retrieval instead of re-reading are sometimes offered as making the question moot. They do not: they are policies, they live in $A(M, b)$, and the model is what says whether a given one is worth its own cost. A compaction pass is itself a model call with a cost c_t, so "compact at 70%" is a decision the backward pass can evaluate rather than a setting that ends the discussion.

What a token-aware net does not give

One resource at a time. The budget of Definition 1 is a scalar, and a deployed system meters several resources at once: tokens, wall-clock, dollars and per-tool quotas, each with its own ceiling and its own replenishment rule. A vector budget is a routine extension, replacing b by $b \in \mathbb{N}^m$ and reading the affordability test componentwise, and Theorem 4 survives it, since every cycle still strictly decreases at least one component, so the state graph is still acyclic and the backward pass still

terminates. What does not survive is the size: the reachable set grows as the product of the component ranges, so the $O(B)$ of the scalar case becomes $O(\prod_i B_i)$ and the exact computation stops being free. Renewable resources are a sharper break. A rate limit that replenishes is not a depleting potential, the acyclicity argument fails outright, and the problem becomes a genuine average-reward control problem rather than a backward pass. That is the boundary of what this paper's machinery buys.

Four things, and the paper is weaker where a reader expects them.

It does not supply the numbers. The costs c_t, the rates λ_t and the correctness probabilities q_ℓ are measurements, and every quantity computed here is only as good as they are. §6 is an attempt to obtain them from published runs rather than to assume them, and Appendix M states what that substitution costs.

It does not model the model. What the agent thought is not a condition anything can depend on; it enters only as an outcome distribution on competing transitions. A net cannot tell you that a cheaper model would have been adequate for *this* task, only what follows once you have said how often it is adequate.

Its central structural result has a hypothesis that can fail. Theorem 4 needs every cycle to contain a cost-bearing transition. A workflow with a free internal loop, such as a retry that consumes nothing or a poll, violates it, and there the acyclicity and the single backward pass are both lost, though the model remains well defined and ordinary value iteration applies.

And it says nothing about renewable resources. Everything here depends on non-renewability, so a rate limit that refills, a context window that can be freed, and a monthly budget observed over many missions are outside the scope as stated. Extending to them is the open problem §7 names first.

H From a real agentic workflow to a token-aware net

The gap between what teams build and what this paper analyzes is closed by a translation, and the translation is mechanical except at four points where it is not. This appendix gives the correspondence, works it on a configuration of the kind engineers actually write, says what to leave out, and ends with a checklist.

THE CORRESPONDENCE

In the system	In the net	The decision it forces
a step of the workflow	a transition, with the place before it holding its precondition	none; this part is bookkeeping
one model call	a transition with $c_t > 0$	what one unit of c_t *is*: a token, a thousand tokens, a cent. Only ratios matter, so pick the coarsest unit that keeps the costs integral
a tool call	a transition whose cost is the arguments *plus the result*	whether the result re-enters context. If it does, and it usually does, the cost belongs to the tool transition and is often the largest in the net
a model tier, or a reasoning-effort setting	competing transitions out of one place with different (c, λ, q)	whether the choice is a race or a policy (§2). A configured tier is a race; a router is an action set
variable-length output	branch splitting with weights $p_i(k)$	how many branches. Three quantiles are usually enough, and E5 shows the upper quantile is the one that decides feasibility
a retry after a failed check	a cycle back to an earlier place, through a transition that still spends	nothing, provided the retry costs something. If it is free, Theorem 4 does not apply
a critic or reflection pass	a transition with its own c_t on the path, plus a branch it enables	whether it is on the critical path for budget as well as for latency; it always is
a sub-agent or delegated task	a subnet whose transitions draw on the same budget place	shared pool or per-principal partition. The model handles both; they are different systems
a compaction or summarization pass	a transition with a cost that does *not* reduce b's consumption downstream	it is a policy action, not a discount. Model it as a call, because it is one
the token or spend ceiling	the initial budget B	none; this is the number the analysis is parameterized by
a context window	a bound on B, via the cost of the calls it permits	whether it is genuinely non-renewable here. If the harness compacts, it is not, and §2 says what survives
a rate limit	*not a budget*: a renewable pool	over a horizon short enough that no replenishment occurs it is a budget; otherwise it is out of scope
an SLA or timeout	not a net element at all: the horizon D at which metrics are evaluated	none, but it interacts: E4 shows budget and deadline are complements

The four rows in which a person has to decide something are the tool-call row, the tier row, the sub-agent row and the context-window row. Everything else is transcription.

WORKED TRANSLATION

Configurations of this shape are what agent frameworks accept:

```
agent:
  model: mid                          → selects among competing transitions
  reasoning_effort: medium            → fixes (c, λ, q) on that one
  max_tokens_per_task: 60000          → the initial budget B
tools:
  - name: search                      → a transition; cost includes the results
    max_results: 8                    → multiplies that cost
  - name: run_tests                   → a transition with a heavy-tailed p_i(k)
verify:
  critic: on                          → a transition on the path, with its own cost
  max_retries: 3                      → bounds a cycle that must still spend
deadline_s: 120                       → the horizon D, not a net element
```

Four observations about this translation, in ascending order of how easily they are missed.

The line that decides the analysis is `max_results: 8`. It reads like a quality setting and it is a budget setting, because eight results re-enter the context and are paid for on every subsequent turn. In the net it multiplies c_{search}, and c_{search} is very often larger than c_{reason}. Nothing in the configuration groups it with `max_tokens_per_task`, and they are the same quantity.

`run_tests` needs a distribution, not a cost. Its output is a stack trace whose size nobody controls, so a single number for c_t is a modelling error rather than a simplification. Split it by the branch construction of §2 into at least a typical and a tail branch, because E5 is precisely the finding that the tail is what decides feasibility.

`max_retries: 3` is a budget policy wearing a reliability costume. Each retry re-runs the reasoning and the critic, so the retry allowance and the token allowance are one decision, and the configuration presents them as two.

Two things appear in the net that appear in no configuration. The first is the terminal action's cost, which must be reserved and which no line reserves; this is the whole content of Theorem 1. The second is the dependence of correctness on spend, $q_i(k)$: the configuration says `reasoning_effort: medium` as though it were a preference, and the model requires it to be a point on a curve.

What not to model

More effort is wasted here than anywhere else. The net is a model of *what is spent and what that permits*, not of cognition, and four things should stay out.

Prompt text and reasoning traces. Their *length* is a cost and their content is not a condition anything can depend on. The content becomes, at most, an outcome probability q.

Tool internals. A search backend is a rate, a cost distribution and an outcome distribution. Its ranking function adds states and answers no question that was asked.

Exact token counts. Only ratios of costs matter to every result in this paper, so a cost model accurate to within a factor that is constant across transitions changes nothing. Chasing exact accounting is the most common way to spend a week and learn nothing.

Anything you will not act on. If no decision depends on distinguishing two steps, merge them into one transition and add their costs. Nets grow fast, and B multiplies the state space, so the discipline matters more here than in an untimed, unbudgeted model.

A checklist

1. Pick the **unit**. Coarse enough that every c_t is a small integer, since B multiplies the state space and only ratios matter.
2. List the **model calls**, and charge each one. A step that calls no model is free and can often be merged away.
3. For every tool call, charge **the result as well as the request**. This is the step most often skipped, and in coding and research agents it is where most of the budget goes.
4. Find the **terminal action** and write down what it costs. Every non-monotonicity in this paper lives in the gap between that cost and what is left when the workflow reaches it.
5. Mark which choices are **configured** and which are **routed**. The first are a race; the second are the action set, and only the second admits a $\pi^\star$.
6. Give the heavy-tailed steps a **distribution**, at least a typical and a tail branch. One number is not a simplification here, it is the wrong model.
7. Check that **every cycle spends**. If one does not, say so: the backward pass of Theorem 4 does not apply and value iteration is needed.
8. Only now attach **rates**, and only if a deadline is in play. Without a deadline the absorption probabilities do not depend on them at all.

Steps 1 to 7 give everything except the deadline-constrained metrics, and they need no latency measurements whatsoever. That is worth knowing before starting: the common assumption that this kind of analysis requires a full performance model of the system is false for three of the four metrics.

Table I2: Four systems, the same chain. Column four is the metric of §3 whose value would actually change a decision in that setting; the others are computable there too and would mostly confirm what the team already believes. Column five is what the policy of Theorem 4 reduces to once it is stated in the system's vocabulary.

System	**The allowance B**	**The dominant spend**	**Metric that bites**	**What $\pi^\star$ says**
coding agent	context window, or session token budget	tool *results*: file reads, test output, stack traces	η vs. F	explore while rich, commit to a patch while still able to pay for it
deep research	spend ceiling per report	retrieved passages, and one full call per source	$B^\star$	widen early, then fund synthesis before the last source
support agent	per-conversation cost cap	conversation history resent each turn	η	escalate to a human when poor, not when stuck
multi-agent crew	a shared organizational quota	inter-agent messages: one output is several inputs	$\bar{C}$ tail	fund the principal on the critical path; cap the others

I FOUR AGENTIC SYSTEMS ON A BUDGET

Table I2 is the summary; the four subsections work the rows out.

The running mission is small enough to work by hand, which is why it is the running mission. This appendix names four systems that are not, and for each one says what the allowance actually is, what competes for it, which of the four metrics would change a decision, and what the optimal policy of Theorem 4 looks like when it is written out in that system's own terms. The last of those is the part worth the pages: in three of the four the policy the backward pass produces is something practitioners already do by instinct, which is a check on the model rather than a novelty, and in the fourth it is something they do not.

1. THE CODING AGENT, AND WHAT A BENCHMARK FAILURE DOES NOT SAY

An agent given a repository and an issue. It searches, reads files, runs tests, edits, re-runs, and submits a patch. The allowance is whichever binds first, the context window or a session token budget, and the two behave the same way in the model because both are a pool that only decreases.

The dominant spend is the one flagged in §1, and it is worth repeating here because it inverts the usual intuition: it is not the model's own reasoning, it is *tool results re-entering context*. A file read costs the file's tokenized length. A test run costs what the runner prints, which for a failing suite is a stack trace of unpredictable size. That has two consequences in the formalism. First, the cost c_t of the *observe* transitions dominates the cost of the *reason* transitions, so the trade-off that decides the mission is exploration against action rather than shallow reasoning against deep. Second, $p_i(k)$ has a heavy upper tail that no reasoning-effort setting controls, which is exactly the regime §5 identifies as dangerous: consumption variance is harmless until its upper tail crosses the feasibility boundary, at which point success collapses. One verbose test run strands a mission that a mean-cost calculation says is comfortably funded. The corresponding design move is to truncate or paginate tool output, and the model says what that move is: it is not a tidiness measure, it is bounding the upper tail of p_i so that the boundary is not crossed.

The metric that bites is the separation of $\mathcal{E}$ from F. Benchmarks in this setting report resolved or unresolved (Jimenez et al., 2024), and an unresolved instance is silent about which of the two happened: the agent's patch was wrong, or the agent never got to a patch because it spent the window reading. These call for opposite fixes, either a better model or a larger window and a stingier exploration policy, and the split is cheap to report, since a harness knows whether it terminated on a submission or on a limit. We would like benchmarks in this setting to report η alongside accuracy, and note that the model makes the reason precise rather than merely urging it: by Theorem 2 the two failure modes have different asymptotics in B, so a system whose failures are mostly F cannot be repaired by budget at all, while one whose failures are mostly $\mathcal{E}$ can.

The policy is recognizable. Backward induction on b gives a threshold structure: while b is large the marginal unit is best spent widening the search, and below a threshold it must be reserved for the edit-and-verify sequence that is the only path into Σ. That is what context compaction, summarization at a watermark, and "stop reading and write the patch" heuristics approximate. The contribution is not the shape but that the threshold is computable from (c_t, λ_t, q_t) rather than tuned.

2. Deep research: the capacity curve is a price list

An agent that answers a broad question by reading many sources and writing a report. B is a spend ceiling per report, and it is unusually explicit here because someone is paying per report and knows it.

The structure is the running mission with the numbers changed. Reading a source is an *observe* whose cost is the passage plus a call; the synthesis is the *act*, and it is expensive, because it re-reads everything gathered. That last clause is what makes the failure mode sharp: the cost of the terminal action *grows with what the earlier steps spent*. A mission that reads sixty sources does not merely have less budget left for synthesis, it has a more expensive synthesis to fund, and the two effects push the same way. In the net this is a cost function $c_{\mathrm{act}}(M, b)$ that depends on the marking. Equation (1) as written charges a constant c_t and permits only the *rates* to depend on (M, b), so this is an extension rather than a special case: replacing c_t by $c_t(M, b)$ leaves Theorems 1 and 4 intact, since acyclicity needs only $c_t(M, b) \geq 1$ on every cycle, but Theorem 2 assumes constant costs (C1) and would need restating. The running mission does not exercise it.

$B^\star$ is the metric here, and it is what a research product can put on a price list: $B^\star(\varepsilon)$ is the least spend at which a report meets a stated reliability, and the curve is what tells a product manager whether the difference between the standard and premium tier is a real difference in achievable quality or only a difference in patience. Theorem 2 is the relevant warning: if the retrieval and reasoning stack has a reliability ceiling below the target, no tier reaches it, and selling a bigger budget is selling something that cannot be delivered.

The policy shape is the least intuitive of the four and, we think, the most useful. It is not "read until the budget is nearly gone". Because the terminal action must be affordable and its cost grows, the backward pass reserves for synthesis first and spends the remainder on breadth, so the number of sources is an output of the budget rather than a parameter set by a prompt.

3. The support agent, and the right reason to escalate

An agent handling a conversation under a per-conversation cost cap. The dominant spend is quiet: the conversation history is re-sent on every turn, so cost grows roughly quadratically in turns while the value of another turn does not.

What makes this vignette structurally different from the other three is that it has an *exit that is cheap in the modeled currency*: escalation to a human. Escalation is a transition with a small c_t, a different and usually higher success probability, and a cost that is real but denominated in something the token budget does not measure. Put it in the net and the backward pass produces a policy with a clean statement: escalate when b falls below the threshold at which the remaining budget can still fund a resolution attempt, which is *not* the same rule as escalating when the agent detects it is stuck.

The difference matters because the confidence rule is the one actually deployed. Escalation is almost always triggered by a confidence or sentiment signal, which is a statement about the semantic state; the model says the trigger should also read the budget state, because a conversation with plenty of allowance left is worth another attempt even when confidence is low, and a conversation near its cap is worth escalating even when confidence is high. The two signals are independent, and one of them is currently unused.

η is again the metric that would change a decision, in a specifically operational way. A support organization sees a deflection rate. It does not see how much of the shortfall is F, meaning the agent could not resolve the issue, and how much is $\mathcal{E}$, meaning the agent hit its cap mid-resolution. Those have opposite remedies and the second is often the cheaper one: raising the cap for a segment is a configuration change, and improving the model is not.

4. THE CREW: A SHARED POOL IS NOT THE SAME AS SHARED FUNDING

A supervisor and several workers drawing on one organizational quota. This is the row §C flags as generalizing least comfortably, and the reason is worth stating properly rather than only flagging.

The model handles a shared pool without any extension, since a shared budget place is a shared budget place. What changes is the meaning of the answer. In the other three systems $\pi^\star$ allocates across *steps of one agent*, and the architect who implements it owns every step. In a crew it allocates across *principals*, so the policy is a statement about which agent is funded to think hard, which is an organizational decision with ownership and governance implications. The mathematics is indifferent to that and the deployment is not.

Two concrete results survive the caveat. First, a shared pool weakly dominates any partition of the same total into per-agent sub-budgets, for the same reason Theorem 5 holds: every partition is a feasible policy in the shared maximization, so the optimum over the shared pool is at least as good. That is an argument for shared quotas. Second, and cutting the other way, the shared pool exposes every principal to every other principal's tail: a worker in a retry loop spends the allowance the critical-path agent needed, and $\bar{C}$'s tail rather than its mean is what determines whether that happens. The design that follows is neither extreme: a shared pool with per-principal caps. Such a scheme leaves the *expected* allocation across principals unchanged, since in the common case no cap binds and the pool is drawn on as before, while bounding what any one principal can take in the rare case that it runs away, which is the case $\bar{C}$'s tail describes. It is exactly the structure a rate-limit configuration already has, arrived at here from the model rather than from experience.

THE COMMON SHAPE

In all four, three statements hold that do not hold of an unbudgeted model of the same system. The mission's terminal action must be *affordable at the moment it is reached*, which is a constraint on everything upstream of it and is the mechanism behind Theorem 1. The failure set splits in two, and the split is invisible to every benchmark listed in the introduction while calling for opposite interventions. And the optimal allocation is a function of remaining budget rather than a fixed setting, computable exactly by one backward pass because spending is irreversible, which is the same irreversibility that made exhaustion expressible in the first place.

J PROOFS

Theorem 2, in full. Monotonicity: the state space of the chain at budget B embeds in that at budget $B' > B$ by $(M, b) \mapsto (M, b + B' - B)$, and every transition enabled in the former is enabled in the image, so every path reaching Σ at budget B has a counterpart at B'; hence $\sigma^\ell_B \le \sigma^\ell_{B'}$. Since the sequence is bounded above by 1 it converges, and its limit is the absorption probability of the unconstrained chain, in which the budget gate never binds. In that chain each reasoning attempt is correct with probability q_ℓ; if incorrect, the verifier fails to catch it with probability $1 - v$, and the mission is then absorbed in F. So each attempt is absorbed in F with probability at least $(1 - q_\ell)(1 - v) > 0$, giving $\bar{\sigma}_\ell \le 1 - (1 - q_\ell)(1 - v) < 1$.

Theorem 4, in full. Consider a directed cycle in the reachable set of the augmented process. Each of its transitions maps (M, b) to $(M', b - c_t)$ with $c_t \ge 0$, so b is nonincreasing along the cycle; returning to the start requires $\sum c_t = 0$, hence $c_t = 0$ for every transition on the cycle. By hypothesis every cycle of the underlying net contains a transition with $c_t \ge 1$, a contradiction. The reachable set is therefore acyclic, and since b strictly decreases across every level, ordering states by b gives a topological order. Backward induction over that order computes the optimal value exactly; each state considers $|A|$ actions and each action inspects a bounded number of successors, giving $O(B\,|P|\,|A|)$. No contraction argument or stopping criterion is required, in contrast to value iteration for a general CTMDP (Puterman, 1994).

Theorem 1, arithmetic. $B = 14$. Observation costs 2, leaving 12. Level H costs 10, leaving 2. Verification costs 1 and action 3, so 4 units are required and 2 remain: the state is in $\mathcal{E}$ by definition, and no transition out of it exists. Hence $\sigma = 0$ exactly. Level M costs 5, leaving $7 \ge 4$, so the mission completes whenever the reasoning was correct: $\sigma = 0.900$. The retry contributes nothing here, and it

is worth saying why, because it is the same arithmetic one level down. A detected error leaves $b = 6$ after verification, and a second M call costs 5 of that 6, leaving 1 against the 4 that verifying and acting still need. Only the adaptive policy, which may answer that retry with L, converts it into the extra 0.042 of §5.

Proposition 1, in full. The first display is the Dvoretzky-Kiefer-Wolfowitz inequality with Massart's constant (Massart, 1990), applied to the random variable Z whose law is a sub-probability on $[0, \infty)$ with the remaining mass at $+\infty$; the inequality concerns $\sup_B |\hat{F}(B) - F(B)|$ over the real line and is insensitive to that atom, since both functions are constant beyond every finite B. For the inversion, write $d = d_n(\alpha)$ and work on the event $\sup_B |\hat{\sigma} - \sigma| \le d$. If $B = \hat{B}^\star(\varepsilon - d)$ then $\hat{\sigma}(B) \ge 1 - (\varepsilon - d)$, hence $\sigma(B) \ge \hat{\sigma}(B) - d \ge 1 - \varepsilon$, so B is feasible for the target and $B^\star(\varepsilon) \le B$. If instead $B < \hat{B}^\star(\varepsilon + d)$ then $\hat{\sigma}(B) < 1 - (\varepsilon + d)$, hence $\sigma(B) \le \hat{\sigma}(B) + d < 1 - \varepsilon$ and B is infeasible, so $B^\star(\varepsilon) \ge \hat{B}^\star(\varepsilon + d)$. Both bounds hold simultaneously for every ε because the event does not depend on ε. Two caveats on using it. The i.i.d. hypothesis treats the benchmark's task set as a sample from a task distribution, which is the reading under which generalization beyond those tasks is meant; read instead as a statement about the fixed 168 tasks, the curve is exact and needs no band at all. And the band is marginal in the configuration: comparisons *between* two configurations evaluated on the same tasks are paired, and the exact McNemar test of §6 is the appropriate instrument there, which is why it is so much sharper than the band.

Theorem 3, in full. Let $\beta_a = \inf\{B : \sigma_a(B) > 0\}$ and $\bar{\sigma}_a = \sup_B \sigma_a(B)$, and suppose $\bar{\sigma}_a < \bar{\sigma}_{a'}$ and $\beta_a < \beta_{a'}$. For $B \in (\beta_a, \beta_{a'})$, the definition of β_a as an infimum gives $\sigma_a(B) > 0$, while $B < \beta_{a'}$ gives $\sigma_{a'}(B) = 0$; the interval is nonempty because $\beta_a < \beta_{a'}$. Put $\delta = (\bar{\sigma}_{a'} - \bar{\sigma}_a)/3 > 0$. Since each curve is nondecreasing and converges to its supremum, there is B_0 with $\sigma_{a'}(B) > \bar{\sigma}_{a'} - \delta$ for $B \ge B_0$, and $\sigma_a(B) \le \bar{\sigma}_a < \bar{\sigma}_{a'} - 2\delta$ for every B; hence $\sigma_a(B) < \sigma_{a'}(B)$ for all $B \ge B_0$. The two displays together are the crossing.

For the necessary condition, suppose the ranking by $\bar{\sigma}$ agrees with the ranking at every budget. If some pair had $\bar{\sigma}_a < \bar{\sigma}_{a'}$ and $\beta_a < \beta_{a'}$, the first display would exhibit a budget at which a strictly beats a', contradicting agreement; so no such pair exists, which is precisely the statement that β is nonincreasing in $\bar{\sigma}$ across the compared set. The converse fails: take σ_a and $\sigma_{a'}$ with the same onset and ordered ceilings but with σ_a rising faster in the middle, so that $\sigma_a > \sigma_{a'}$ on an interior interval while both the onset and the ceiling orderings agree. The count of crossing pairs follows because each pair satisfying the hypothesis contributes at least one crossing and the pairs are distinct.

Theorem 5, in full. Dominance: each constant policy π_ℓ is an element of the feasible set of the maximization solved by Theorem 4, so $\sigma_B^{\pi^\star} \ge \sigma_B^{\pi_\ell}$ for every B. The capacity statement follows because σ_B is nondecreasing in B under each fixed level (Theorem 2), so the sublevel set defining $B^\star$ is an up-set and a pointwise dominating curve reaches any target no later.

Margin: let r satisfy the three hypotheses, and consider the (generally suboptimal) policy ρ that plays ℓ at the first reasoning step and ℓ'' at the second. On the event that the first attempt is correct, ρ and π_ℓ coincide and earn the same value. On the event that it is wrong and the verifier catches it, which has probability $(1 - q_\ell)v$, the state is $(M_r, r - c_\ell - c_{\mathrm{ver}})$. From there π_ℓ attempts ℓ again with $r - c_\ell - c_{\mathrm{ver}} < c_\ell + \kappa$ and so earns exactly 0 by Theorem 1, whereas ρ attempts ℓ'' with $r - c_\ell - c_{\mathrm{ver}} \ge c_{\ell''} + \kappa$ and earns $\sigma^{\ell''}_{r - c_\ell - c_{\mathrm{ver}}}$. On the remaining event the two again coincide. Hence $\sigma^\rho_r - \sigma^{\pi_\ell}_r = (1 - q_\ell)v\, \sigma^{\ell''}_{r - c_\ell - c_{\mathrm{ver}}}$, and $\sigma^{\pi^\star}_r \ge \sigma^\rho_r$ gives the claim. In the running mission at $r = 12$ the three hypotheses hold for $\ell = \mathrm{M}$, $\ell'' = \mathrm{L}$, and the bound evaluates to $0.1 \times 0.6 \times 0.7 = 0.042$, which §5 finds is the exact gap: the inequality is tight whenever the cheaper retry is itself unimprovable.

K THE STREAMING VARIANT

The body charges a block of tokens when a call completes. Inference is closer to a stream, and the model accommodates this without new machinery. Let reasoning consume budget one unit at a time through a transition t_{token} with rate μ, competing against a completion transition t_{done} whose hazard $h(k)$ depends on the number k of units already spent:

$$(\text{reasoning}, k, b) \xrightarrow{\ \mu\ } (\text{reasoning}, k+1, b-1), \qquad (\text{reasoning}, k, b) \xrightarrow{\ h(k)\ } (\text{done}, k, b).$$

Now k is reasoning effort and the agent is drawing on an intelligence stream until it has enough to stop. Correctness on completion is $q(k)$, increasing and concave in the usual case. The decision at each state is whether to continue drawing or to act, which is the explicit stochastic form of *think more versus act now*, and Theorem 4 still applies because t_{token} strictly decreases b.

The qualitative findings survive: a policy that always thinks to the largest k exhausts the budget on missions whose downstream steps then cannot fire. We do not claim a direction for how the optimal stopping threshold on k moves with the remaining budget. Two effects oppose each other, since a richer agent can afford to think longer but also has more downstream value to protect, and which dominates depends on h and on the cost of the remaining steps; the threshold is computed by the same backward induction rather than asserted. The block-charging model of the body is the special case in which h is a point mass, and we use it in the experiments because it keeps the state space small enough to enumerate exactly.

L THE EXACT EXPERIMENTS OF §5: PROTOCOL

What is in this appendix. The full sweeps behind §5 are Table L3 (success against budget, and the capacity curve it inverts to) and Table L4 (the deadline interaction, E4).

E1 TO E5 IN FULL

The optimal level is a function of the budget (E1 to E3). Figure L2 sweeps B against the three fixed levels: three regimes and so two changes of ordering. Below 11 units only the cheap model is affordable at all and returns 0.700 while both others return exactly 0; between 11 and 15 the mid level is best; from 16 the deep level is. The optimal policy tracks this, choosing L for a remaining budget in $[2, 8]$, M in $[9, 13]$ and H from 14, and its thresholds are the quantities of Theorem 1, $c_{\text{M}} + \kappa = 9$ and $c_{\text{H}} + \kappa = 14$. The policy is a lookup table, not a heuristic, and it says something a practitioner would not otherwise guess: reason cheaply when poor, not because the cheap model is good, but because an expensive answer you cannot act on is worth nothing. At $B = 14$ adaptivity gains 0.042 over the best fixed level, by a path opposite to what one might assume: observation leaves $b = 12$, where $\pi^\star$ selects M rather than H, leaving 7, which is enough to verify and act. The gain is entirely the retry, since with probability 0.1 the reasoning is wrong, with probability 0.6 the verifier catches it, and the second attempt faces $b = 6$, where $\pi^\star$ drops to L and succeeds with probability 0.7: that is $0.1 \times 0.6 \times 0.7$, the bound of Theorem 5 attained exactly, and a fixed M policy cannot make that second attempt at all. Inverting the curves (Table L3, Appendix L) shows where adaptivity is worth anything, which is easy to overclaim: at the obvious target 0.90 there is *no* capacity gap, since $\pi^\star$ and the fixed mid level both reach it at $B = 11$; adaptivity saves a third only against always choosing the *most accurate* level, which needs 16. The strict gap appears at 0.92, where no fixed level arrives before 16 and $\pi^\star$ arrives at 14; where one level is unambiguously right it ties, which is correct behaviour and worth reporting, since adaptivity is insurance against the budget moving rather than free performance. The ceilings are the other half: the cheap model never attains 90% and the mid model never attains 98% at any budget whatsoever, at 0.854 and 0.957, matching Corollary 1 to every digit reported. That is the answer to "can we just give it more tokens?".

Deadlines and variance (E4, E5). Two further sweeps are in Appendix L. Budget and time are complements, not substitutes: at $D = 8$ raising B from 16 to 30 buys 0.008, while at $D = 40$ raising B from 12 to 16 buys 0.080. And consumption variance is not monotone in its effect: holding the mean cost of level H at 10 and spreading it symmetrically at $B = 20$, $D = 25$, a spread of ± 2 leaves σ at 0.980, ± 4 *raises* it to 0.986 because the cheap half frees budget, and ± 6 collapses it to 0.496 with $\eta = 0.500$. The reason is structural: variance is irrelevant while the whole distribution stays inside the feasible region and catastrophic once its upper tail crosses the boundary at which the remaining budget can no longer fund the action. So the mean token cost of a step says nothing about mission risk without the tail, a prediction §6 then measures on real logs.

Implementation. A compact Python model of the token-augmented chain. Absorption probabilities and the optimal policy come from the backward induction of Theorem 4, exact up to floating point. Deadline-constrained quantities use uniformization with the Poisson series truncated at $N = \Lambda D + 10\sqrt{\Lambda D} + 60$ terms. There is no sampling anywhere, so no error bars are reported, but

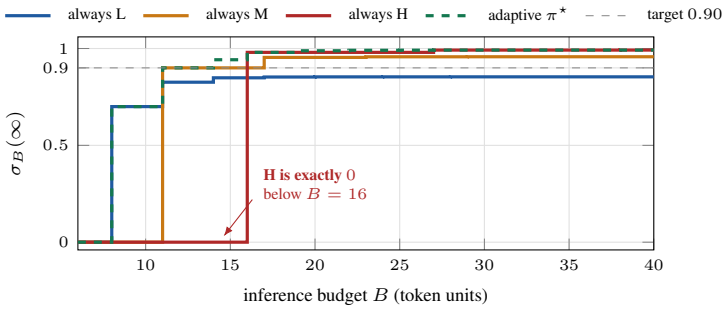

Figure L2: **The capacity picture, computed.** Each fixed level is unusable below the budget it needs, rises in steps as successive retries become affordable, and is capped by a ceiling no budget crosses (0.854, 0.957, 0.992). The most accurate level is the *worst* choice for every $B < 16$, returning exactly zero (Thm. 1), and the adaptive policy is the upper envelope and more, strictly exceeding every fixed level at $B = 14$ and $B \geq 19$.

Table L3: **The model, computed exactly** by backward induction. *Top*: mission success $\sigma_B(\infty)$; the most accurate level is the worst possible choice for $B \leq 15$ and returns *exactly* zero, because it cannot leave enough budget to act on its own conclusion. *Bottom*: the capacity curve $B^\star(\varepsilon)$; "never" means no budget suffices because the level's ceiling lies below the target (Thm. 2). **Best fixed** is the honest baseline for Thm. 5: adaptivity buys nothing at 0.90, two units at 0.92, nothing at 0.95 and above. Against the *habit* of always choosing the most accurate level it would save 31% at 0.90, which is a statement about the habit.

	always L	always M	always H	best fixed	**adaptive $\pi^\star$**
success $\sigma_B(\infty)$ at budget B					
$B = 10$	0.7000	0.0000	0.0000	0.7000 (L)	**0.7000**
$B = 12$	0.8260	0.9000	0.0000	0.9000 (M)	**0.9000**
$B = 14$	0.8487	0.9000	0.0000	0.9000 (M)	**0.9420**
$B = 16$	0.8487	0.9000	0.9800	0.9800 (H)	**0.9800**
$B = 20$	0.8535	0.9540	0.9800	0.9800 (H)	**0.9884**
$B = 24$	0.8536	0.9572	0.9800	0.9800 (H)	**0.9908**
$B = 32$	0.8537	0.9574	0.9918	0.9918 (H)	**0.9919**
ceiling	0.8537	0.9574	0.9919	0.9919	**0.9919**
capacity $B^\star(\varepsilon)$ for target $1 - \varepsilon$					
0.90	never	11	16	11	**11**
0.92	never	17	16	16	**14**
0.95	never	17	16	16	**16**
0.98	never	never	16	16	**16**

"exact" would be the wrong word for the deadline-constrained quantities and we do not use it for them. Two error terms remain. Floating point, throughout. And the truncation, which is bounded by the discarded Poisson tail: since the transient probabilities are a convex combination weighted by $\text{Poisson}(\Lambda D)$ masses, the omitted mass $\sum_{n>N} e^{-\Lambda D}(\Lambda D)^n/n!$ bounds the error in absolute value. Over the whole grid the paper reports, the largest such tail is at $B = 30$, $D = 40$, where $\Lambda = 4$ and $N = 346$, and it is 1.4×10^{-37}. That is far below the accumulated floating-point error and far below the four decimals the tables print, so the truncation is not the binding inaccuracy anywhere; it is reported because "no error at all" would be false. The net definition, the exact analysis and its raw output accompany the submission as `model_exact.py` and `results_model.json`.

Parameters. As tabulated in §2: costs $c_{\text{obs}} = 2$, $c_{\text{ver}} = 1$, $c_{\text{act}} = 3$; rates $\lambda_{\text{obs}} = 2$, $\lambda_{\text{ver}} = 4$, $\lambda_{\text{act}} = 1$; verifier detection $v = 0.6$. Reasoning levels (c, λ, q) equal to $(2, 1.0, 0.70)$, $(5, 0.667, 0.90)$ and $(10, 0.4, 0.98)$. Budgets are integers, so the augmented chain has $8(B + 1)$ states, at most 488 for the largest budget reported.

Table L4: E4 in full: token-constrained success $\sigma_B(D)$ under both constraints. Reading down a column shows budget helping, reading across a row shows time helping, and the two are complements rather than substitutes. At $D = 8$ the mission is time-limited and raising B from 16 to 30 buys 0.008; at $D = 40$ the same increase buys 0.012 and raising B from 12 to 16 buys 0.080. Additional tokens have little value when the deadline is the binding constraint, and the model says which regime a system is in.

	$B = 12$	$B = 16$	$B = 20$	$B = 30$
$D = 8$	0.8782	0.8881	0.8951	0.8961
$D = 12$	0.8984	0.9613	0.9694	0.9720
$D = 20$	0.9000	0.9792	0.9876	0.9910
$D = 40$	0.9000	0.9800	0.9884	0.9919

Table M5: The corpus. AppWorld publishes the raw trajectories of its own leaderboard entries: four agent architectures crossed with four models, every configuration run on the same tasks, each run logging the code the agent wrote and every byte the environment returned. It does not log tokens; the bill is reconstructed from those logs and from the prompt-assembly code shipped in the same repository, and validated against a corpus that does log them (Table M10). All trajectories were taken from AppWorld's published logs. We did not rerun the agents, and the original policies did not observe a token budget.

what was read	**how much**
configurations (architecture × model)	14
trajectories	8,190
turns	212,550
tool output produced	245.4M tokens
agent code produced	16.8M tokens
charged input, as run	**1.37B tokens**
the same without the harnesses' trimming	9.96B tokens (7.3×)
scored a success	1,277
scored a failure, having concluded (error)	3,752
scored a failure, never having concluded	3,161

On the choice of parameters. The three levels are meant to caricature a real menu, a small model, a mid model, and a deep reasoning mode, with costs in the ratio $1 : 2.5 : 5$ and error rates falling by roughly a factor of three at each step. The qualitative results depend only on the ordering and on the non-reasoning steps costing enough that the deep level can starve them; they do not depend on the specific numbers, and the accompanying code sweeps them.

M THE BENCHMARK OF §6: PROTOCOL AND FURTHER RESULTS

What is in this appendix. The corpus as read is Table M5; the reconstruction's validation against provider-reported usage is Table M10, with the per-call and cumulative shapes drawn in Figures M3 and M4. The undocumented run-length ceilings behind §6's remark on declared caps are Table M11; the measured capacity curves and their inversion are Tables M6 and M7; the non-conclusion split is Table M8 and the two escalation triggers Table M9. E13's cost-correctness dependence is Table M13, E14's trimming counterfactual Table M12, and the harder split's ceilings Table M14.

What was read, and from where. AppWorld's published leaderboard outputs (`appworld download experiment-outputs`) and its source repository, which carries the agent code and prompt files that fix how each architecture assembles a prompt; and, for the validation of §6, the OpenManus logs in MAST's `traces/` tree. Both are public corpora, read here for a quantity their own authors did not report. Nothing in the benchmark calls a model or needs an API key, so every number below is reproducible from the released archives alone.

Table M6: **Measured capacity curves** on AppWorld `test_normal`: σ_B is the share of tasks that both succeeded and cost at most B tokens; the last column is the published leaderboard score, which is the right-hand limit of the curve. The ordering by score is *not* the ordering under a budget: the top configuration returns exactly zero below 23k while at 10k the third-ranked one returns 0.339. Every crossing is between two entries of the same leaderboard. Pointwise 95% Clopper-Pearson half-widths are at most 0.078 ($n = 168$).

configuration	$10k$	$30k$	$100k$	$300k$	$1.0M$	**ceiling**
ReAct/GPT-4o	0.000	0.042	0.351	0.488	0.488	0.488
Plan+Exec/GPT-4o	0.000	0.000	0.113	0.423	0.446	0.446
FullCode/GPT-4o	0.339	0.339	0.339	0.339	0.339	0.339
Plan+Exec/GPT-4-turbo	0.000	0.000	0.167	0.327	0.327	0.327
FuncCall/GPT-4o	0.000	0.173	0.244	0.310	0.321	0.321
FuncCall/GPT-4-turbo	0.000	0.161	0.250	0.304	0.304	0.304
ReAct/GPT-4-turbo	0.000	0.077	0.238	0.268	0.268	0.268
FullCode/GPT-4-turbo	0.238	0.256	0.256	0.256	0.256	0.256
FullCode/Llama-3-70B	0.220	0.244	0.244	0.244	0.244	0.244
ReAct/Llama-3-70B	0.000	0.018	0.179	0.208	0.208	0.208
FullCode/DeepSeek-33B	0.131	0.131	0.131	0.131	0.131	0.131
Plan+Exec/Llama-3-70B	0.000	0.000	0.036	0.083	0.089	0.089
ReAct/DeepSeek-33B	0.000	0.036	0.071	0.071	0.071	0.071
Plan+Exec/DeepSeek-33B	0.000	0.000	0.012	0.012	0.018	0.018

Table M7: **The capacity curve inverted**: $B^\star(\varepsilon)$, the least allowance attaining completion $1 - \varepsilon$ on `test_normal`. **Leaderboard pick** is the configuration chosen from the score alone; **cheapest** is the one reaching the same target for least. The saving is not adaptivity within a mission but the difference between choosing on accuracy and choosing on capacity. Budgets are rounded; savings are computed from the unrounded thresholds.

target	**leaderboard pick**	**cheapest**	**which**	**saving**
0.10	41k	4k	FullCode/GPT-4o	89%
0.20	51k	4k	FullCode/GPT-4o	91%
0.30	64k	7k	FullCode/GPT-4o	88%
0.40	119k	119k	ReAct/GPT-4o	n/a

The leaderboard pick is **ReAct/GPT-4o** (ceiling 0.488). It returns zero below $B = 23k$; some configuration is already positive at $B = 4k$.

The accounting model. One environment interaction is one model call for the ReAct, plan-and-execute and whole-program architectures; the function-calling architecture is excluded from per-call statistics because one assistant message may carry several tool calls and the log flattens them. Charged input at turn k is the instruction header plus the surviving history, where "surviving" means after simulating the harness's own trimming rule on the character lengths that rule is written in: `react_controller.py` rebuilds the thread and, while it exceeds `max_output_length` characters, blanks the earliest observation outside the last five messages and then drops messages off the front. The simulation walks the same two pointers in the same order, so the observations it keeps are the observations the model saw. One approximation is stated rather than hidden: the controller measures the task statement together with the thread and the published logs do not carry that string, a few hundred characters against a 20,000-character cap. Tokens are counted with `o200k_base`, which is what the GPT-4o runs were billed against; the Together-hosted models use their own vocabularies, which is why every comparison in §6 is drawn within one tokenizer and why the headline numbers are tokens rather than dollars.

Prices. Where dollars appear they use provider list prices per million tokens at the time the runs were published (GPT-4o \$5/\$15, GPT-4-turbo \$10/\$30, and the two Together-hosted open models at their size-tier rates), and the price table is a module constant a reader who disagrees with a rate can change. No conclusion in §6 depends on it.

Table M8: E10, metric η measured as far as the logs allow, on `test_normal`. An agent that called `complete_task` reached a conclusion and was wrong; one that never called it did not conclude. The scorer records both as a failure. **Proxy** is the bit's own check against the only ground truth available, that a scored success must have concluded, and it is 1.000 on every configuration and on all 8,190 trajectories. **At cap** is the share of the non-concluded runs that stopped at exactly the largest turn count the configuration ever reached, which is the strongest limit-termination evidence these artifacts support; the complement is not evidence of anything, it is where the logs are silent. **Crossover** is the allowance above which errors outnumber non-conclusions.

configuration	succ	error	unfin.	unf/fail	at cap	proxy	crossover
Plan+Exec/DeepSeek-33B	3	10	155	0.939	0.01	1.000	never
ReAct/DeepSeek-33B	12	33	123	0.788	0.84	1.000	never
Plan+Exec/Llama-3-70B	15	79	74	0.484	0.22	1.000	594k
Plan+Exec/GPT-4o	75	51	42	0.452	0.07	1.000	353k
FullCode/DeepSeek-33B	22	83	63	0.432	0.84	1.000	11k
FullCode/Llama-3-70B	41	86	41	0.323	0.29	1.000	11k
ReAct/GPT-4o	82	64	22	0.256	1.00	1.000	158k
Plan+Exec/GPT-4-turbo	55	87	26	0.230	0.04	1.000	210k
FullCode/GPT-4o	57	86	25	0.225	0.96	1.000	8k
ReAct/GPT-4-turbo	45	97	26	0.211	0.08	1.000	94k
FullCode/GPT-4-turbo	43	109	16	0.128	0.81	1.000	7k
ReAct/Llama-3-70B	35	117	16	0.120	0.94	1.000	103k
FuncCall/GPT-4o	54	103	11	0.096	0.00	1.000	61k
FuncCall/GPT-4-turbo	51	109	8	0.068	0.00	1.000	48k

Over all 8,190 published trajectories: 1,277 successes, 3,752 errors and 3,161 non-conclusions, so **45.7% of this benchmark's recorded failures are runs that never produced an answer**. Of those, 1,300 (41.1%) stopped at their configuration's observed ceiling; for the other 1,861 the published logs do not name a cause, so they are counted as non-concluded and not as token exhaustion.

Reproducing. The reconstruction described above runs against AppWorld's published trajectory logs and the prompt-assembly code shipped in that project's repository. We do not redistribute either, so the pipeline is specified here rather than shipped: the accounting model, the tokenizer used per model family, the validation corpus and its R^2 values, and the filters behind every count above are all stated in this appendix. What the supplementary material does contain is `stats_bench.py`, which recomputes the statistical claims of §6 from the per-configuration shares printed in Table M6 and therefore needs no corpus: the Clopper-Pearson intervals, the uniform band of Proposition 1, the exact McNemar test at $B = 10$k, and the reordering of Figure 2(b). The 41 ordered crossings are computed on the full replay curves rather than on the five grid points printed here, so that count is reported above but is not among the quantities the script recomputes. Thirty-eight offline tests (`test_tb_corpus.py`) pin the parser, the trimming simulation, the three-way outcome rule and the policy arithmetic against hand-written fragments, and run without the data.

Table M9: E12, two escalation triggers on the same pair. Every configuration ran every task, so a two-stage policy over them is evaluated exactly rather than sampled: run FullCode/GPT-4-turbo, then spend what is left on ReAct/GPT-4o. **Confidence** escalates when the first stage failed to declare itself finished. **Budget** ignores that signal and escalates whenever the remainder covers the second stage. **Oracle** escalates when the first stage was in fact wrong and is a bound, not a policy. In all three, when the second stage runs its answer replaces the first stage's; that is what makes the budget column fall from 0.500 at 200k to 0.488 at 500k, because a larger allowance makes escalation affordable on tasks the cheap stage had already solved, and the expensive stage then answers them wrongly. Naive escalation is not monotone in the budget, which is itself worth knowing. *Worth* in the text means $\max_B$ of the policy's success minus the better of the two fixed columns at the same B: 0.000 for confidence, which never leads, and 0.077 for budget, at $B = 100k$. **Escalated** is how often the confidence trigger fires at all, which is why it is worth so little: the cheap stage declares itself finished on 90.5% of tasks and is wrong on 71.7% of those. The same pair is the best of 182 ordered pairs, so the gain is not a search artifact.

allowance B	FullCode/GPT-4-turbo	ReAct/GPT-4o	confidence	budget	oracle	escalated
5k	0.167	0.000	0.167	**0.167**	0.167	0.000
10k	0.238	0.000	0.238	**0.238**	0.238	0.000
15k	0.256	0.000	0.256	**0.256**	0.256	0.006
25k	0.256	0.018	0.256	**0.256**	0.256	0.065
50k	0.256	0.196	0.256	**0.321**	0.339	0.095
100k	0.256	0.351	0.268	**0.429**	0.458	0.095
200k	0.256	0.482	0.274	**0.500**	0.542	0.095
500k	0.256	0.488	0.274	**0.488**	0.548	0.095
1.0M	0.256	0.488	0.274	**0.488**	0.548	0.095

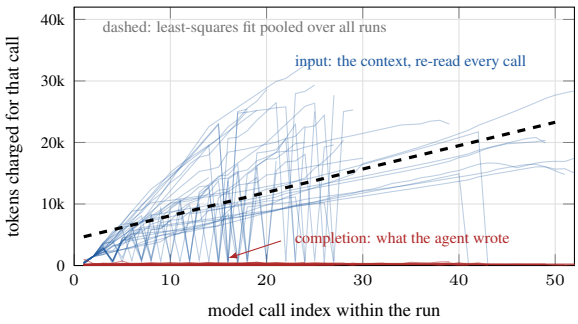

Figure M3: E6 drawn: every model call of 30 OpenManus GAIA runs, as the provider's own usage report records it. Each faint blue line is one run's input side and each red line the same run's completion side. Input rises with the call index because the context carries everything the tools have printed so far, so the bill of an n-call run grows like n^2; completion does not rise and never leaves the floor. Summed over all 940 calls, completion is **1.40%** of what was charged. The reasoning trace is the part of the bill a thinking-budget parameter controls, and it is the part that does not matter. The dashed line is a least-squares fit pooled over all runs, $+380$ tokens per call; it is shallower than the $+619$ Table M10 reports because that one is the median of the within-run slopes, and pooling mixes the growth inside a run with the differences between runs in how much their tools print.

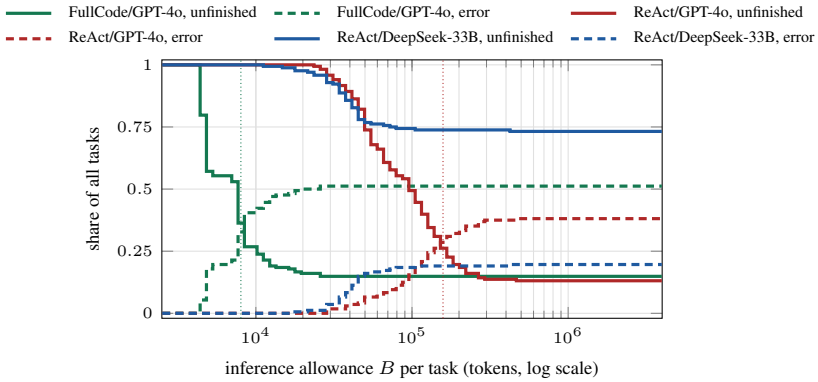

Figure M4: E10 and E11 drawn, on `test_normal`. Solid: the share of tasks that produced no answer under allowance B, either because the bill exceeded B or because the run never concluded within it. Dashed: the share that ended in an error, an agent that concluded and was wrong. The first falls with the allowance and the second rises to a ceiling, so the two cross at most once, and the dotted line marks where. Left of a configuration's crossover the useful purchase is capacity; right of it, a better agent. The whole-program agent crosses at 8k tokens and ReAct on GPT-4o at 158k, an order of magnitude apart on the same tasks; ReAct on DeepSeek never crosses at all. Only the over-budget part of the solid curve is attributable to tokens by construction; the rest is non-conclusion whose cause the logs do not name, which is why the curve is not labelled exhaustion.

Table M10: E6, the accounting model against a corpus that records the provider's own usage report. OpenManus logs `Token usage: Input=..., Completion=...` beside every model call, for 30 GAIA runs published to study reasoning failures. The *completion share* row is the claim of Appendix I that reasoning is not what a tool-using agent spends its allowance on, tested directly. The *last against second* row is the size of the error in charging a transition a constant c_t, and its sign: a constant cost understates late steps, so it understates the risk of running out. It is measured against the second call rather than the first because the first carries only the system prompt, with no tool result in it yet.

what the provider's own counter says	
model calls with a usage report	940
per-call input, growth per call (median within-run)	+619 tokens
runs with a positive slope	30/30
cumulative bill vs. calls, R^2 linear / quadratic	0.795 / 0.879
completion tokens as a share of the bill	**1.40%**
last call against second, same run (median ratio)	13.2×
in tokens	1,496 → 19,949
runs stopped by their step cap	4/30

Table M11: The declared allowance against the one that actually stopped the runs, on `test_normal`. AppWorld's configuration files write the allowance down in four different currencies (model calls, dollars, steps, characters), and none of them is reported beside the score. In three of the four architectures the declared counter is never reached and the runs pile up at a lower ceiling instead, whose accounting rule is not recoverable from what was published: a model call that returns no executable code is billed and leaves no trace in the log.

configuration	declared	median	p95	observed cap	share there
FullCode/DeepSeek-33B	10	3	7	7	0.327
FullCode/GPT-4-turbo	10	2	7	7	0.083
FullCode/GPT-4o	10	3	7	7	0.155
FullCode/Llama-3-70B	10	3	7	7	0.095
FuncCall/GPT-4-turbo	15	12	25	50	0.006
FuncCall/GPT-4o	15	13	41	92	0.006
Plan+Exec/DeepSeek-33B	100	96	97	98	0.006
Plan+Exec/GPT-4-turbo	100	18	46	96	0.006
Plan+Exec/GPT-4o	100	22	94	97	0.018
Plan+Exec/Llama-3-70B	100	20	97	97	0.095
ReAct/DeepSeek-33B	100	71	71	71	0.613
ReAct/GPT-4-turbo	100	12	63	71	0.012
ReAct/GPT-4o	100	16	71	71	0.131
ReAct/Llama-3-70B	100	16	71	71	0.101

Table M12: E14, what bounding tool output is worth, on `test_normal`. Two of the four architectures cap the history they carry and two do not, so the counterfactual needs no simulation: each configuration's bill is recomputed with the cap removed. The *median* moves little; the mean and the tail move by orders of magnitude, and the last column is the ratio of totals, so it is the factor by which the mean moves. That separation is the paper's point about p_i: a truncation rule is a tail-bounding measure, and the mean token cost of a step says nothing about mission risk without it.

configuration	median	median†	p99	p99†	total†/total
Plan+Exec/Llama-3-70B	145k	159k	913k	115.3M	23.6×
Plan+Exec/GPT-4o	171k	223k	858k	25.4M	6.9×
ReAct/GPT-4o	94k	98k	510k	12.8M	5.8×
ReAct/Llama-3-70B	79k	84k	487k	1.1M	1.8×
ReAct/GPT-4-turbo	63k	63k	472k	1.4M	1.7×
Plan+Exec/GPT-4-turbo	128k	144k	736k	2.0M	1.7×
ReAct/DeepSeek-33B	415k	478k	507k	2.3M	1.5×
Plan+Exec/DeepSeek-33B	821k	1.1M	872k	1.7M	1.5×
FullCode/GPT-4o	7k	7k	31k	31k	1.0×
FuncCall/GPT-4o	46k	46k	791k	791k	1.0×
FuncCall/GPT-4-turbo	38k	38k	302k	302k	1.0×
FullCode/GPT-4-turbo	5k	5k	28k	28k	1.0×
FullCode/Llama-3-70B	8k	8k	25k	25k	1.0×
FullCode/DeepSeek-33B	7k	7k	26k	26k	1.0×

† with the harness's own history cap removed.

Table M13: E13, the model's independence assumption tested. §2 splits a transition into a consumption branch and a correctness branch and treats them as independent given the level. On `test_normal` they are not: on every configuration the runs that succeeded were cheaper than the runs that did not. $\mathbb{P}(\text{cheaper})$ is the chance a random success cost less than a random failure, which is distribution-free and so is not read off these heavy tails by a correlation coefficient. The consequence: the model fitted with measured (p_i, q) predicts $\sigma_B = q\,\mathbb{P}(\text{cost} \le B)$ and *understates* the empirical curve by up to 0.096, so the abstraction errs toward over-provisioning.

configuration	median, success	median, failure	ratio	$\mathbb{P}$(cheaper)
FullCode/DeepSeek-33B	4k	11k	2.52×	0.900
ReAct/DeepSeek-33B	29k	424k	14.88×	0.857
FullCode/GPT-4o	4k	8k	1.72×	0.823
FullCode/Llama-3-70B	4k	11k	2.45×	0.820
ReAct/GPT-4o	53k	137k	2.57×	0.787
ReAct/GPT-4-turbo	38k	80k	2.09×	0.769
ReAct/Llama-3-70B	49k	99k	2.00×	0.736
Plan+Exec/GPT-4o	139k	212k	1.52×	0.718
FullCode/GPT-4-turbo	4k	5k	1.05×	0.682
Plan+Exec/Llama-3-70B	116k	157k	1.36×	0.643
Plan+Exec/GPT-4-turbo	98k	152k	1.55×	0.638
FuncCall/GPT-4o	28k	47k	1.67×	0.611
FuncCall/GPT-4-turbo	25k	44k	1.74×	0.598

Table M14: E7 and E8 repeated on the harder split: capacity curves on AppWorld `test_challenge`: σ_B is the share of tasks that both succeeded and cost at most B tokens. The last column is each configuration's published leaderboard score, which is the right-hand limit of its curve. The ordering by that score is *not* the ordering under a budget: the top configuration returns exactly zero below $B = 29k$, while at $B = 10$k the third-ranked one already returns 0.165. Every crossing here is between two entries of the same leaderboard.

configuration	$10k$	$30k$	$100k$	$300k$	$1.0M$	**ceiling**
ReAct/GPT-4o	0.000	0.002	0.158	0.302	0.302	0.302
Plan+Exec/GPT-4o	0.000	0.000	0.034	0.177	0.197	0.197
FullCode/GPT-4o	0.165	0.192	0.192	0.192	0.192	0.192
FuncCall/GPT-4o	0.000	0.065	0.161	0.180	0.180	0.180
ReAct/GPT-4-turbo	0.000	0.019	0.115	0.175	0.175	0.175
FuncCall/GPT-4-turbo	0.000	0.058	0.146	0.146	0.146	0.146
FullCode/GPT-4-turbo	0.120	0.125	0.125	0.125	0.125	0.125
Plan+Exec/GPT-4-turbo	0.000	0.000	0.046	0.110	0.110	0.110
FullCode/Llama-3-70B	0.060	0.070	0.070	0.070	0.070	0.070
FullCode/DeepSeek-33B	0.058	0.058	0.058	0.058	0.058	0.058
ReAct/Llama-3-70B	0.000	0.005	0.029	0.034	0.034	0.034
ReAct/DeepSeek-33B	0.000	0.000	0.029	0.029	0.029	0.029
Plan+Exec/Llama-3-70B	0.000	0.000	0.012	0.022	0.024	0.024
Plan+Exec/DeepSeek-33B	0.000	0.000	0.005	0.007	0.007	0.007

N GLOSSARY AND SYMBOL TABLE

Two vocabularies meet in this paper and each has a word the other uses differently. The worst offender is *token*: in Petri-net writing it is a dot in a place, and in language-model writing it is the unit a budget is denominated in. This paper uses **mark** for the dot and reserves *token* for the model's unit throughout, so that "a token budget" means what a practitioner expects and never anything else.

Terms.

Term	**Meaning here**
place	a circle: a condition that holds or a resource that is counted
mark	a dot in a place; one instance of that condition or one unit of that resource
marking	the list of how many marks sit in each place. The entire workflow state
transition	a bar: an event. A step of the workflow, usually a model or tool call
firing	the event happening: marks are *removed* from input places and added to output places
enabled	every input place holds enough marks *and* the budget covers the cost
incidence matrix	$C = Post - Pre$: what firing each transition does to each place
budget	the remaining allowance b, carried in the state alongside the marking
token cost	c_t: what firing t draws from the budget. Zero for steps that call no model
absorbing state	one with no outgoing transitions; the process stays once it arrives
exhaustion	absorption in $\mathcal{E}$: the workflow has moves and the budget affords none
capacity curve	$B^\star(\varepsilon)$: the least allowance meeting a reliability target
policy	a rule choosing an action set from the state, here from (M, b)
backward induction	solving for the optimal policy by working from the poorest states up
uniformization	the method by which finite-horizon probabilities are computed exactly

Symbols.

Symbol	Meaning
P, T	places and transitions
$Pre, Post, C$	arc multiplicity matrices and the incidence matrix $C = Post - Pre$
M	a marking; M_0 the initial one
b	remaining budget; B the initial budget, so the state is (M, b) started at (M_0, B)
c_t	the token cost of transition t
$\lambda_t(M, b)$	the rate of t; it may depend on the state
$T_{\mathrm{En}}(M, b)$	the transitions that are both workflow-enabled and affordable, (1)
$\mathfrak{B}$	a budget-augmented stochastic Petri net, Definition 1
$\Sigma, F, \mathcal{E}$	the absorbing classes: success, error, and token exhaustion
$\sigma_B(D)$	σ_B, token-constrained success within the deadline
$\eta(B, D)$	η, the probability of ending specifically by running out
$\bar{C}(B, D)$	$\bar{C}$, expected consumption
$B^\star(D, \varepsilon)$	$B^\star$, the capacity curve
q_ℓ, $q_i(k)$	correctness of reasoning level ℓ; correctness as a function of spend
$p_i(k)$	the distribution of units consumed by call i
$A(M, b)$, π, $\pi^\star$	the action set, a policy, and the optimal policy
D	the deadline. Not a net element: the horizon at which metrics are evaluated

Two phrases used with a technical meaning. *Structural* describes a claim that holds for every admissible assignment of rates, because it follows from what is enabled rather than from how fast anything is; an unaffordable transition being absent from the state space is structural. *Exact* describes a number obtained by solving a linear system or summing a uniformization series to a stated tolerance, rather than by simulating trajectories: an exact quantity carries no sampling error and cannot be improved by running longer.