Polybee AI Research 2026-11-09 · Version 1.5.2 (Submission Candidate [cs.AI])

Polybee F3-Swarm v1.5.2: Verifier-Guided Heterogeneous SLM Routing, Symbolic Evidence Normalization, and Controlled Memory-Pipeline Benchmarks on Commodity Edge Hardware

Henrique Oliveira[1,*]

[1]Polybee AI Research Laboratory, São Paulo, SP, Brazil *Corresponding author (https://polybee.ai/)

Open-Source Code & Telemetry Repository: https://github.com/henrique-polybee/polybee-f3-swarm

Abstract

Deploying autonomous agentic workflows on commodity consumer hardware (16 GiB RAM at $> 80\%$ operating system occupancy, leaving < 2.8 GiB of free physical RAM) exposes a sharp trade-off between memory capacity, latency, and reliability. Executing medium-to-large models (7B–14B) locally triggers severe NVMe SSD paging, whereas restricting execution to sub-billion Small Language Models (SLMs, e.g., 0.5B parameters) introduces systematic failure modes: positional logit bias toward the first enumerated option, numeric/monetary formatting errors in localized Brazilian Portuguese (PT-BR) extraction, and fragile code synthesis. We present **Polybee F3-Swarm v1.5.2**, an experimental study and prototype systems framework organized into two complementary tracks across five laboratories (Labs 01–05) on an Intel Core i7-11370H CPU (4C/8T, AVX-512): (1) **Track A — Controlled Memory-Pipeline Benchmark (Labs 01–02):** Under hard Win32 per-process working-set quotas (`SetProcessWorkingSetSizeEx` at 600 MiB) and unbuffered NVMe SSD I/O (`FILE_FLAG_NO_BUFFERING`), we compare streaming a 24-layer dense matrix transformation pipeline (4096×4096, 1.50 GiB `float32`) from disk against partitioning the same matrices across three resident-weight worker processes communicating via local TCP loopback (`127.0.0.1`) with software-emulated network RTTs. Resident-weight TCP pipelining eliminates 33.00 GiB (33,792 MiB) of SSD reads across 32 steps and achieves a **13.70× throughput speedup at** 0.4 **ms emulated LAN RTT** (19.79 tok/s, TTFT 95.33 ms, TPOT 50.54 ms vs. 1.44 tok/s local SSD streaming), **10.82× at** 4.5 **ms Wi-Fi RTT** (15.62 tok/s), and **3.35× at** 60.0 **ms WAN RTT** (4.84 tok/s) with exact IEEE-754 parity ($\max|\Delta| = 0.0$). (2) **Track B — Sequential-Lifecycle Heterogeneous SLM Cascade (Labs 03–05):** Using real quantized GGUF models via `llama-server` under a single-active-server RAM budget (< 1.5 GiB for Tiers 1–2), we orchestrate `Qwen2.5-0.5B-Instruct` (Tier-1), `Qwen2.5-1.5B-Instruct` (Tier-2), and `Qwen2.5-Coder-3B-Instruct` (Tier-3) across $N = 30$ PT-BR operational tasks. Combining positional logit debiasing ($T = 1.6, \beta_A = 2.5$), option-order inversion checks, Controlled Micro-CoT with 1-token suffix closure, deterministic substring-anchored monetary normalization (**Motor B**), and four-case functional unit testing ($K = 4$) with counterexample-guided repair (**AACR**), **Policy 4 (Swarm v1.0)** achieves **76.7% accuracy** (23/30, 95% Wilson CI [59.1%, 88.2%]) in **84.70 s** (2,019.4 J estimated at 24 W), outperforming standalone 1.5B (**73.3%**, 111.18 s, 2,635.9 J) in both accuracy and speed (-23.8% latency). Adding Tier-3 repair in **Policy 5** trades $+7.1\%$–9.0% higher latency (119.07–121.19 s) for higher task completion: **90.0%** (27/30, CI [74.4%, 96.5%]) in the unpatched diagnostic run (**P5a**) and **96.7%** (29/30, CI [83.3%, 99.4%]) in-sample after correcting a benchmark oracle typo in `T26` and applying suffix closure (**P5b**). To audit out-of-sample generalization, we conduct a post-hoc adversarial audit of Class 3 functions across 32 unseen test cases ($M_{\text{HO}} = 4$ per task): across the 7 tasks with verbatim code strings serialized in the historical JSON logs (`precise_tier3_results.json` and `lab05_tier3_closure_results.json`), F3-Swarm achieves **96.4% (27/28) case-level accuracy** (case false-acceptance rate $\hat{\epsilon}_{\text{FA}}^{\text{case}}(4) = 1/28 = 3.57\%$) and **85.7% (6/7) strict task-level accuracy** (task failure rate $\hat{\epsilon}_{\text{FA}}^{\text{task}}(4,4) = 1/7 = 14.3\%$); including the reconstructed Lab 04 function for `T29` yields **96.9% (31/32) case-level** ($\hat{\epsilon}_{\text{FA}}^{\text{case}}(4) = 3.125\%$) and **87.5% (7/8) task-level accuracy** ($\hat{\epsilon}_{\text{FA}}^{\text{task}}(4,4) = 12.5\%$), corresponding to a **held-out-adjusted 30-task aggregate of 93.3% (28/30, 95% Wilson CI** [78.7%, 98.2%]**)** when combined with prior Class 1 (11/12) and Class 2 (10/10) outcomes.

Keywords: Heterogeneous SLM Cascades, Edge Inference, Uncertainty Calibration, Micro-Chain-of-Thought, Rashomon Set Multiplicity, Counterexample-Guided Code Repair, Neuro-Symbolic Verification, Memory-Bounded Systems.

1 Introduction and Scope of Claims

Deploying language model agents in cost-sensitive operational environments confronts both economic and hardware constraints. For organizations operating in emerging-market currencies (such as the Brazilian Real, BRL), routing

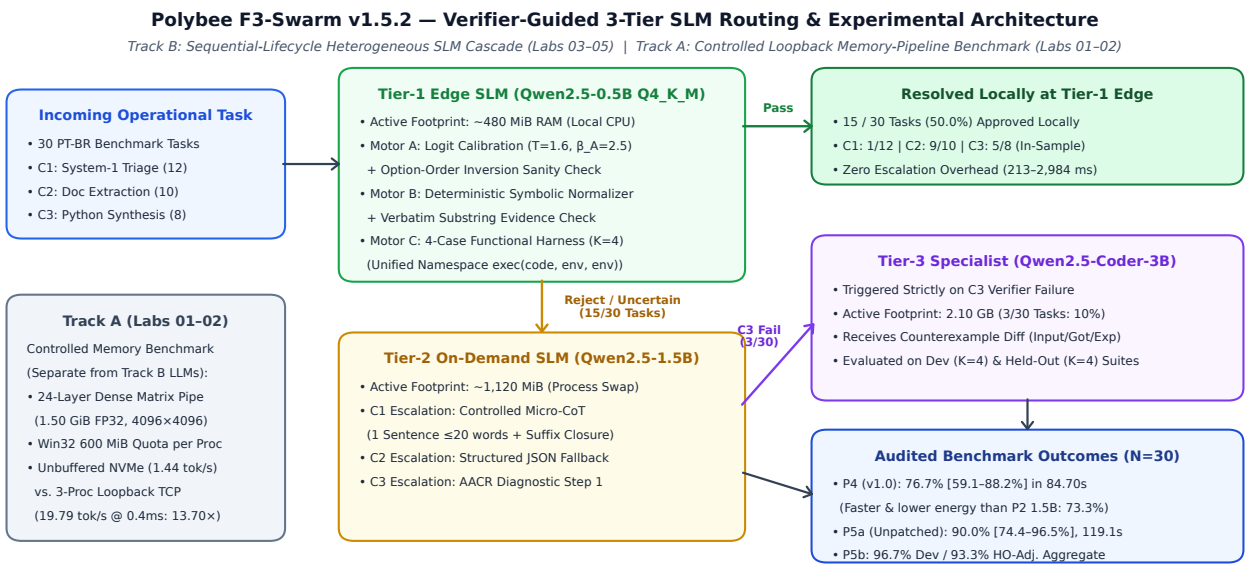


Figure 1 | Overview of Polybee F3-Swarm v1.5.2 and Explicit Two-Track Experimental Design. Track B (Labs 03–05, right) orchestrates three real quantized GGUF language models sequentially on local CPU (`Qwen2.5-0.5B`, `Qwen2.5-1.5B`, and `Qwen2.5-Coder-3B`) using calibrated System-1 routing, substring-anchored symbolic normalization (Motor B), and a four-case functional execution harness ($K = 4$) with counterexample-guided AACR escalation. **Track A (Labs 01–02, bottom-left)** independently evaluates OS-level memory-hierarchy trade-offs on the same host by comparing unbuffered NVMe SSD weight streaming against a 3-process resident-weight TCP loopback pipeline under hard 600 MiB Win32 working-set quotas and software-emulated RTTs.

high-volume administrative triage, financial extraction, and data-cleaning workflows exclusively to commercial cloud APIs incurs substantial recurring costs. However, migrating these workloads onto existing commodity office laptops (16 GiB total DDR4 RAM running Windows 11 and standard corporate software) encounters a strict physical memory ceiling: background operating system and application processes routinely occupy $> 80\%$ of physical memory (~ 13.0 GiB), leaving fewer than 2.8 **GiB of free physical RAM** (Section 4.1).

When a practitioner attempts to run a monolithic 7B–14B model (4.5–8.5 GiB) inside this < 2.8 GiB envelope, the operating system pages weight tensors to NVMe SSD storage, collapsing autoregressive throughput. Conversely, restricting inference to sub-billion parameter Small Language Models (SLMs, such as `Qwen2.5-0.5B-Instruct` [11], occupying ~ 480 MiB of RAM) avoids paging but exposes three domain-level failure modes observed in our experiments:

1. **Positional First-Option Bias in System-1 Triage:** In multiple-choice operational classification (Class 1), the uncalibrated 0.5B model assigns $> 91\%$ softmax probability to option `'A'` on tasks where factual evidence mandates option `'B'` (`T06`, `T10`, `T11`), defeating naive confidence-threshold cascades.
2. **Localized Monetary Formatting Fragility:** In Brazilian Portuguese (PT-BR) financial extraction (Class 2), 0.5B models frequently emit ambiguous thousand/decimal strings (e.g., `"32.400.00"` or unquoted `14.850.20`) that break standard JSON parsers despite extracting the exact literal evidence span from the source text.
3. **Verbosity Truncation and Finite-Test Multiplicity in Code Synthesis:** In Tier-2 (1.5B) escalations, unconstrained JSON Chain-of-Thought (CoT) [14] truncates before closing the JSON object (`T05`), whereas pure 1-token decoding suppresses needed intermediate deduction (`T01`, `T08`). In Python function synthesis (Class 3), both 0.5B and 1.5B models fail on multi-branch formatting and check-digit rules (`T23`, `T25`, `T26`), and functions that pass a small finite set of unit tests ($K = 4$) may still fail on unseen boundary inputs—a practical manifestation of the *Rashomon Effect* [8, 9, 7].

Explicit Delineation of Experimental Scope. To maintain strict scientific precision and avoid conflating systems simulations with end-to-end distributed deployments, we explicitly separate our investigation into two distinct experimental tracks (Table 1):

- **Track A (Memory-Hierarchy and Weight-Movement Benchmark — Labs 01 & 02):** Evaluates the OS-level mechanics of memory overflow on a single Intel Core i7-11370H host. Lab 02 compares

unbuffered NVMe SSD weight streaming (`FILE_FLAG_NO_BUFFERING`) against a three-process resident-weight pipeline connected via local TCP loopback (`127.0.0.1`) under hard 600 MiB Win32 working-set quotas (`SetProcessWorkingSetSizeEx`) and software-injected RTT delays (`time.sleep`) over a 24-layer dense matrix transformation workload (4096 × 4096, 1.50 GiB `float32`). *Scope boundary:* Lab 02 is a controlled single-host benchmark of resident-weight TCP pipelining versus disk weight streaming; it does not implement a full autoregressive LLM (with multi-head attention, KV-cache, or token embeddings) across separate physical computers.

- **Track B (Verifier-Guided Heterogeneous SLM Cascade — Labs 03, 04 & 05):** Evaluates real quantized GGUF models (`Qwen2.5-0.5B-Instruct`, `Qwen2.5-1.5B-Instruct`, and `Qwen2.5-Coder-3B-Instruct`) via `llama-server.exe` (`llama.cpp` [12]) on $N = 30$ PT-BR operational tasks. *Scope boundary:* Track B enforces memory limits via sequential single-server process lifecycle management (≤ 1 model loaded in local RAM at any instant), rather than executing `Qwen2.5-Coder-3B` across the Track A TCP matrix pipeline. Integrating multi-host `llama.cpp` RPC sharding with the Track B agentic cascade is designated as future work.

Table 1 | Positioning of Polybee F3-Swarm v1.5.2 relative to recent distributed inference and SLM routing frameworks. ✓ denotes a capability implemented and empirically evaluated in this study; *Sim/Loop* denotes controlled single-host loopback/emulated evaluation.

System / Study	Multi-Host Network	OS Quota Swap Benchmark	SLM Uncertainty Routing	Positional Debiasing + Flip	Deterministic PT-BR Normalizer	Code Repair + Held-Out Audit
Petals (Borzunov et al., 2023 [2])	✓ (WAN P2P)	✓	×	×	×	×
Faster / Speculative Cascades (Narasimhan et al., 2024 [5])	×	×	✓ (Deferral Rule)	×	×	×
FlowSpec (Liu et al., 2025/2026 [4])	✓ (Edge Testbed)	×	✓ (Tree Speculative)	×	×	×
SWARM-LLM (Dahshan et al., Apr 2026 [3])	✓ (Edge + Cloud FM)	×	✓ (Uncertainty/Safety)	×	×	×
System One / Jev (TypeSafe AI, Sep 2026 [6])	×	×	✓ (Typed System-1)	✓	×	×
F3-AGI v4.0 (Oliveira, Oct 2026 [1])	×	×	×	×	✓ (Motor B)	✓ (ARC DSL)
Polybee F3-Swarm v1.5.2 (This Work)	*Sim/Loop* (Track A)	✓ (Win32 600 MiB)	✓ (Track B 3-Tier)	✓ ($T = 1.6, \beta_A = 2.5$)	✓ (Substring-Anchored)	✓ ($K = 4$ Dev + $K = 4$ HO)

2 Related Work (2022–2026)

Collaborative and Pipelined Distributed Inference. The observation that pooling memory across multiple nodes can outperform local disk offloading was established at scale by **Petals** [2], which demonstrated pipeline-parallel inference and fine-tuning of 176B-parameter models (`BLOOM-176B`) over the Internet, outperforming local RAM/SSD offloading on consumer GPUs. More recently, **FlowSpec** [4] introduced continuous pipelined tree-based speculative decoding for distributed edge LLM inference, combining score-based step-wise verification with dynamic draft expansion to achieve 1.37 × –1.73× speedups on distributed testbeds. In Track A (Labs 01–02), we do not claim to invent distributed weight hosting; rather, we provide a reproducible, OS-instrumented micro-benchmark on Windows 11 (`SetProcessWorkingSetSizeEx` and `FILE_FLAG_NO_BUFFERING`) quantifying the exact crossover point between unbuffered NVMe SSD streaming and resident-weight TCP pipelining on an AVX-512 laptop CPU.

Small Language Model Swarms and Speculative Cascades. Routing queries across models of increasing capacity has been studied extensively through model cascades and deferral rules. **Faster Cascades via Speculative Decoding (Speculative Cascades)** [5] characterized optimal deferral rules that interleave small draft models and larger target models, outperforming standalone cascades and standard speculative decoding on Gemma and T5 benchmarks. Closest in spirit to our Track B architecture is **SWARM-LLM: Collaborative Inference for Edge-based Small Language Models** (Dahshan et al., April 2026 [3]), which coordinates three heterogeneous edge-hosted SLMs with an optional 70B cloud foundation model using lightweight uncertainty estimates, safety/policy gating, and uncertainty-weighted peer consensus, limiting cloud invocations to roughly 25% of queries. Similarly, on September 15, 2026, TypeSafe AI introduced **Jev** [6], a non-autoregressive *System One model* trained to emit typed, calibrated software decisions orders of magnitude faster than conversational text generation.

How F3-Swarm v1.5.2 Complements Existing Literature. While SWARM-LLM [3] and Speculative Cascades [5] route primarily on intrinsic uncertainty or token-level speculative acceptance, operational workflows in enterprise administration require *extrinsic domain verification*. Extending our neuro-symbolic **F3-AGI v4.0** architecture [1] and addressing the predictive multiplicity (Rashomon effect) analyzed by Breiman [8], Fisher et al. [9], and Mlodozeniec and Frei [7], F3-Swarm v1.5.2 couples an ultra-lightweight edge SLM (0.5B) to: (i) explicit first-option logit debiasing with option-order inversion checks, (ii) a deterministic PT-BR monetary normalizer anchored by literal substring containment, and (iii) counterexample-guided Python code repair (**AACR**) audited against both in-sample ($K = 4$) and unseen held-out ($K_{\text{HO}} = 4$) functional test suites under a single-active-process memory ceiling.

3 Analytical Models and Conditional Formulations

To ensure that our theoretical claims match the exact scope of our experimental evidence, we state all analytical relationships with explicit conditional hypotheses and error terms.

3.1 Memory-Hierarchy Latency Model for Resident-Weight Pipelining (Track A)

Consider an L-stage sequential matrix transformation pipeline with total weight footprint $M_{\text{model}} = \sum_{l=1}^{L} m_l$ bytes executing under a per-process physical RAM working-set quota $Q_{\text{RAM}} < M_{\text{model}}$. In single-process streaming (Arm 1), at most $L_{\text{RAM}} = \lfloor Q_{\text{RAM}}/m_l \rfloor$ stages remain resident in DRAM (bandwidth B_{DRAM}), while the remaining $\Delta M_{\text{swap}} = M_{\text{model}} - L_{\text{RAM}} m_l$ bytes are read from unbuffered NVMe storage (bandwidth B_{SSD}) at each step t:

$$\text{TPOT}_{\text{stream}}(M_{\text{model}}, Q_{\text{RAM}}) = T_{\text{compute}}(L) + \frac{L_{\text{RAM}} m_l}{B_{\text{DRAM}}} + \frac{\Delta M_{\text{swap}}}{B_{\text{SSD}}} + L_{\text{swap}}\, \tau_{\text{io}}, \tag{1}$$

where τ_{io} is the per-layer OS seek/read syscall overhead. Conversely, when the L stages are partitioned across $K = \lceil M_{\text{model}}/Q_{\text{RAM}} \rceil$ processes (Arm 2) such that each shard $M_{\text{model}}/K \le Q_{\text{RAM}}$ remains resident in DRAM ($\Delta M_{\text{swap}} = 0$), step t incurs zero disk I/O and transmits K intermediate activation vectors of size $S_{\text{act}} = 4\, d_{\text{model}}$ bytes over TCP with per-hop latency τ_{hop} and bandwidth B_{net}:

$$\text{TPOT}_{\text{pipe}}(M_{\text{model}}, K) = T_{\text{compute}}(L) + \frac{M_{\text{model}}}{B_{\text{DRAM}}} + K\left(\tau_{\text{hop}} + \frac{S_{\text{act}}}{B_{\text{net}}}\right). \tag{2}$$

Proposition 1 (Analytical Crossover Condition for Resident-Weight Pipelining vs. Disk Streaming)**.** *Under the additive latency model of Equations (1)–(2) with identical CPU compute time $T_{compute}(L)$, resident-weight K-stage pipelining achieves lower per-step latency than single-process disk streaming ($TPOT_{pipe} < TPOT_{stream}$) if and only if the per-hop communication delay satisfies:*

$$\tau_{hop} \;<\; \tau_{crit} \;:=\; \frac{\Delta M_{swap}}{K}\left(\frac{1}{B_{SSD}} - \frac{1}{B_{DRAM}}\right) + \frac{L_{swap}}{K}\,\tau_{io} - \frac{S_{act}}{B_{net}}. \tag{3}$$

For the Lab 02 parameters ($L = 24$, $m_l = 64\,MiB$, $L_{RAM} = 8$, $\Delta M_{swap} = 1,024\,MiB$, $K = 3$, $B_{SSD} = 1,603.1\,MB/s$, $S_{act} = 16\,KiB$), the disk streaming penalty is $\Delta M_{swap}/B_{SSD} \approx 638.8\,ms/step$, yielding $\tau_{crit} \approx 212\,ms$ per hop. Furthermore, because IEEE-754 floating-point matrix operations are executed in identical deterministic order on lossless TCP byte streams, $\max_j |\mathbf{h}_{L,j}^{pipe} - \mathbf{h}_{L,j}^{stream}| = 0.0$.

3.2 Positional Logit Calibration and Conditional Order-Inversion Detection (Motor A)

In Class 1 multiple-choice triage over options $\mathcal{O} = (o_1, \ldots, o_K)$ indexed by tokens $\{A, B, \ldots\}$, Tier-1 (`Qwen2.5-0.5B`) produces raw logits $\mathbf{z}(\mathcal{O})$. Standard temperature scaling $\tilde{z}_k = z_k/T$ preserves the argmax ordering ($\arg\max_k z_k/T = \arg\max_k z_k$) and therefore cannot alter a positionally biased top-1 choice. We apply **Prior-Penalized Temperature Calibration**:

$$\tilde{z}_k(\mathcal{O}) = \frac{z_k(\mathcal{O}) - \beta_k}{T}, \qquad p_k(\mathcal{O}) = \frac{\exp\big(\tilde{z}_k(\mathcal{O})\big)}{\sum_{j=1}^{K} \exp\big(\tilde{z}_j(\mathcal{O})\big)}, \qquad (\beta_A = 2.5,\;\; \beta_{k\neq A} = 0,\;\; T = 1.6), \tag{4}$$

coupled with an **Order-Inversion Sanity Gate** comparing the semantic option $\Phi(\mathcal{O})$ selected under forward ordering $\mathcal{O}$ against $\Phi(\pi(\mathcal{O}))$ under reversed ordering $\pi(\mathcal{O}) = (o_K, \ldots, o_1)$:

$$\mathcal{G}_{\text{Sys1}}(x; \mathcal{O}) = \mathbb{I}\Big(\max_k p_k(\mathcal{O}) \ge \tau_{\text{conf}}\Big) \;\wedge\; \mathbb{I}\big(\Phi(\mathcal{O}) = \Phi\big(\pi(\mathcal{O})\big)\big), \qquad (\tau_{\text{conf}} = 0.85). \tag{5}$$

Lemma 1 (Conditional Detection of Additive First-Slot Bias under Option Inversion)**.** *Assume an additive decomposition $z_k(\mathcal{O}) = s(o_k) + b_k$, where $s(o_k)$ is the order-invariant semantic score returned by the model and $b_A = b_{pos} > 0$ ($b_{k\neq A} = 0$) is a first-slot positional bias. In a binary choice $\mathcal{O} = (o_1, o_2)$ where $s(o_2) - s(o_1) = \Delta s > 0$:*

1. *If residual slot bias dominates the semantic margin ($b_{pos} - \beta_A > \Delta s > 0$), then $\Phi(\mathcal{O}) = o_1$ under forward ordering but $\Phi(\pi(\mathcal{O})) = o_2$ under reversed ordering, so $\Phi(\mathcal{O}) \neq \Phi(\pi(\mathcal{O}))$ and $\mathcal{G}_{Sys1}(x; \mathcal{O}) = 0$ (positionally induced false positives are detected and escalated).*

2. ***Limitation (Semantic Error Floor $\epsilon_{sem} > 0$):*** *If the model holds an order-invariant semantic misconception such that $s(o_{wrong}) - s(o_{true}) > |b_{pos} - \beta_A|$, then $\Phi(\mathcal{O}) = \Phi(\pi(\mathcal{O})) = o_{wrong}$. Thus, order inversion detects positional slot bias but does* not *guarantee detection of semantic errors (as observed empirically in task `T08` on Tier-2 1-token decoding, where $\Phi(\mathcal{O}) = \Phi(\pi(\mathcal{O})) = \texttt{C} \neq \texttt{A}$).*

3.3 Controlled Micro-CoT, Suffix Closure, and Substring-Anchored Normalization

When $\mathcal{G}_{\text{Sys1}}(x; \mathcal{O}) = 0$, Tier-2 (`Qwen2.5-1.5B`) executes a **Controlled Micro-CoT** prompt instructing the model to generate one concise justification sentence (≤ 20 words) followed by `"Resposta final: <LETRA>"`. In our unpatched Lab 05 run (**Policy 5a**, $N_{\text{predict}} = 45$), this recovered `T01` (22 words) and `T08` (11 words), but truncated on `T03` (31 words) and `T09` (28 words) before emitting the final label. In **Policy 5b**, if a trace $r_{1:m}$ reaches the token budget ($N_{\text{predict}} = 65$) without emitting `"Resposta final:"`, a **Deterministic Suffix Closure** appends `"\nResposta final: "` to $x \oplus r_{1:m}$ and queries 1 token ($T = 0.0$):

$$\hat{y}_{\text{Tier2}} = \Psi_{\text{sem}}\left(\arg\max_{v \in \mathcal{V}} P_{\theta_{1.5\text{B}}}\left(v \mid x \oplus r_{1:m} \oplus \texttt{"\textbackslash nResposta final: "}\right)\right). \tag{6}$$

For Class 2 (PT-BR Financial Extraction from document $\mathcal{D}$), **Motor B** (`FastSymbolicNormalizer`) sanitizes multi-dot numeric strings $\mathcal{N}_{\text{BRL}}(\texttt{valor_raw})$ and applies the **Substring-Anchored Extraction Check**:

$$\mathcal{V}_{\text{MotorB}}(y; \mathcal{D}) = \mathbb{I}\big(\texttt{evidencia} \subseteq_{\text{literal}} \mathcal{D}\big) \wedge \mathbb{I}\big(|\texttt{evidencia}| \geq 5\big) \wedge \mathbb{I}\big(\mathcal{N}_{\text{BRL}}(\texttt{valor_raw}) > 0\big) \wedge \mathbb{I}\big(\mathcal{R}_{\text{doc}}(\texttt{doc_id}) = 1\big). \tag{7}$$

Scope note: $\mathcal{V}_{\text{MotorB}} = 1$ guarantees that the evidence span is a verbatim substring of $\mathcal{D}$ and that the monetary value and CPF/CNPJ syntax are well-formed; if a document contains multiple distinct monetary figures, $\mathcal{V}_{\text{MotorB}}$ verifies format and substring presence rather than semantic role disambiguation.

3.4 Four-Case Functional Verification, Counterexample AACR, and Finite-Test Risk Bounds

In Class 3 (Python Function Synthesis), each task provides a specification S and a finite development test suite $\mathcal{T}_{\text{dev}} = \{(x_i, y_i^*)\}_{i=1}^{K}$ with $K = 4$ input-output test cases.[1] Candidate code h is parsed via `ast.parse` and executed in a **Unified-Namespace Execution Harness**:

$$\texttt{exec}\big(\text{AST_Clean}(h),\ \mathcal{E}_{\text{exec}},\ \mathcal{E}_{\text{exec}}\big), \qquad \mathcal{V}_K(h; \mathcal{T}_{\text{dev}}) = \mathbb{I}\left(\frac{1}{K}\sum_{i=1}^{K} \mathbb{I}\big(h(x_i) = y_i^*\big) = 1.0\right). \tag{8}$$

Security Disclaimer (Non-Isolated Prototype Harness): Passing a single dictionary $\mathcal{E}_{\text{exec}}$ as both `globals` and `locals` resolves CPython's module-import scoping quirk (allowing top-level `import re` and `from datetime import datetime` statements to remain visible inside `def solucao(...)`). However, because $\mathcal{E}_{\text{exec}}$ exposes Python's standard `__builtins__`, Equation (8) is *not* a secure security sandbox against untrusted or adversarial code. Production deployments must isolate execution inside OS-level containers, `seccomp` filters, microVMs, or WebAssembly (WASI) runtimes.

When $\mathcal{V}_K(h^{(k)}; \mathcal{T}_{\text{dev}}) = 0$, the harness constructs the counterexample diff $\Delta_{\text{AACR}}(h^{(k)}) = \{(i, x_i, y_i^*, h^{(k)}(x_i)) \mid h^{(k)}(x_i) \neq y_i^*\}$ and escalates to the next tier.

Proposition 2 (Conditional Cascade Accuracy Bound under Finite-Test Verification)**.** *Let $\mathcal{V}_K(h) \in \{0, 1\}$ be a finite-test functional verifier over K test cases, and let $p_1 = \Pr(\mathcal{V}_K(h_{T1}) = 1)$, $p_2 = \Pr(\mathcal{V}_K(h_{T2}) = 1 \mid \mathcal{V}_K(h_{T1}) = 0)$, and $p_3 = \Pr(\mathcal{V}_K(h_{T3}) = 1 \mid \mathcal{V}_K(h_{T1}) = 0, \mathcal{V}_K(h_{T2}) = 0)$.*

1. ***In-Sample Acceptance Rate:*** *The probability that the 3-tier cascade produces a candidate passing all K development tests is exactly:*

$$Pass_K(\mathcal{S}_{F3}) = 1 - (1 - p_1)(1 - p_2)(1 - p_3) \ \geq \max_{m \in \{1,2,3\}} Pass_K(\textit{Tier-}m). \tag{9}$$

[1] In earlier laboratory logs (`lab03`–`lab05`), this 4-case functional unit test check was labeled `Rashomon LOO-CV`; throughout this audited manuscript, we use the exact computer-science term **Four-Case Functional Unit Testing** ($K = 4$) and reserve out-of-sample generalization claims strictly for our unseen Held-Out Test Suite $\mathcal{T}_{\text{HO}}$ (Section 5.2).

2. ***Out-of-Sample Generalization Bounds (Case-Level ϵ_{FA}^{case} vs. Task-Level ϵ_{FA}^{task}):*** *Because $K < \infty$ tests cannot prove correctness over an infinite domain $\mathcal{X}$, we distinguish two post-verification error quantities on an unseen evaluation suite $\mathcal{T}_{HO} = \{(x_j^{HO}, y_j^{HO})\}_{j=1}^{M_{HO}} \sim \mathcal{D}_{OOS}^{M_{HO}}$:*

$$\epsilon_{FA}^{case}(K) := \Pr_{(x,y^*)\sim\mathcal{D}_{OOS}} \big(h(x) \neq y^* \;\big|\; \mathcal{V}_K(h;\mathcal{T}_{dev}) = 1\big) \quad \textit{(Per-Case Error Rate)}, \tag{10}$$

$$\epsilon_{FA}^{task}(K, M_{HO}) := \Pr_{\mathcal{T}_{HO}\sim\mathcal{D}_{OOS}^{M_{HO}}} \left(\bigvee_{j=1}^{M_{HO}} \big(h(x_j^{HO}) \neq y_j^{HO}\big) \;\middle|\; \mathcal{V}_K(h;\mathcal{T}_{dev}) = 1 \right) \leq M_{HO}\, \epsilon_{FA}^{case}(K). \tag{11}$$

Then expected case-level accuracy and strict task-level pass rate (M_{HO}/M_{HO}) on unseen inputs satisfy:

$$Acc_{OOS}^{case}(\mathcal{S}_{F3}) \geq Pass_K(\mathcal{S}_{F3})\big(1 - \epsilon_{FA}^{case}(K)\big), \qquad Pass_{OOS}^{task}(\mathcal{S}_{F3}) \geq Pass_K(\mathcal{S}_{F3})\big(1 - \epsilon_{FA}^{task}(K, M_{HO})\big). \tag{12}$$

4 Track A Experimental Results: Hardware Inventory and Controlled Memory-Pipeline Benchmark (Labs 01 & 02)

4.1 Lab 01: Hardware Telemetry and Parametric Analytical Projections

In **Lab 01** (`collect_hardware_inventory.py`, `hardware_inventory_lab01.json`), Win32 kernel inspection (`GlobalMemoryStatusEx`, `GetLogicalProcessorInformationEx`, and CPUID Leaf 7) on our test host established:

- **CPU & Vector ISA:** Intel Core i7-11370H (4 physical cores / 8 logical threads @ 3.30 GHz, Tiger Lake H35, 35 W nominal TDP) with verified hardware support for `AVX2`, `AVX-512F`, `AVX-512DQ`, `AVX-512BW`, and `AVX-512VL`.
- **Physical RAM Envelope:** Out of 15.788 GiB total physical RAM, baseline OS and developer processes occupied 12.999 GiB (82.0% load), leaving 2.789 **GiB of free physical RAM**.
- **Unbuffered NVMe SSD Throughput:** Sequential reads measured 1,452.95 MB/s (1.42 GB/s) in Lab 01 and 1,603.13 MB/s (1.57 GB/s) in Lab 02 under `FILE_FLAG_NO_BUFFERING`.

Using these hardware constants, Lab 01 (`f3_swarm_lab01_results.json`) first computed *analytical parametric projections* across five hypothetical network profiles (LAN 0.5 ms, Wi-Fi 5.0 ms, Fiber 12.0 ms, Regional WAN 25.0 ms, Intercontinental WAN 60.0 ms), motivating the controlled OS-level experiment in Lab 02.

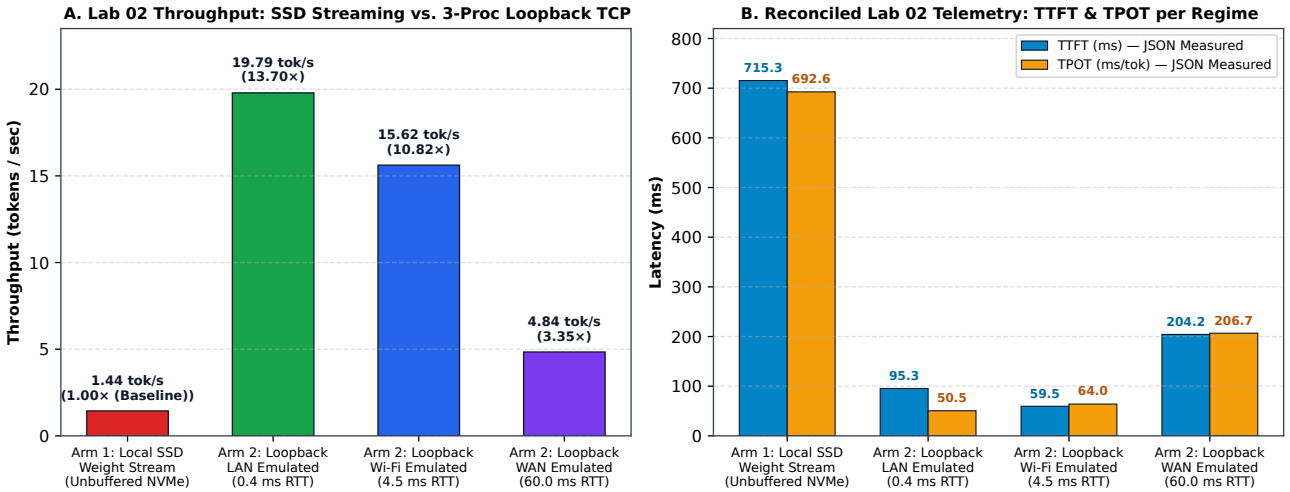
4.2 Lab 02: Controlled OS Working-Set Quota and Loopback TCP Pipeline Benchmark

In **Lab 02** (`f3_swarm_lab02_physical_test.py`, `lab02_physical_results.json`), we instantiated a 24-layer dense matrix transformation pipeline ($d_{\text{model}} = 4096$, 64 MiB per layer = 1.50 GiB total in `float32`, executing matrix projections, layer normalization, and non-linear activations across 32 sequential steps). All processes were constrained by a hard Win32 kernel working-set cap of 600 **MiB** via `SetProcessWorkingSetSizeEx(..., QUOTA_LIMITS_HARDWS_MAX_ENABLE)`.

- **Arm 1 (Local Unbuffered SSD Weight Streaming):** A single process kept 8 layers (512 MiB) resident in RAM and streamed the remaining 16 layers (1,024 MiB) from NVMe SSD at every step using `FILE_FLAG_NO_BUFFERING`.
- **Arm 2 (3-Process Resident-Weight TCP Loopback Pipeline):** Three separate Python worker processes (`PID 6208`, `22960`, `19272`) ran concurrently on the same host, each holding 8 layers (512 MiB weights; measured working sets 529.5 MiB, 530.6 MiB, and 546.8 MiB) in physical RAM and exchanging 16 KiB activation vectors over `127.0.0.1` TCP sockets with `time.sleep`-emulated RTTs for three regimes: **LAN (0.4 ms RTT)**, **Wi-Fi (4.5 ms RTT)**, and **WAN (60.0 ms RTT)**.
- **Arm 3 (Simulated Orchestrator Primitives):** Evaluated a 64 MiB linear System-1 triage filter (0.704 ms/item across 30 prompts), Distributed Speculative Decoding (DSD) over the loopback pipeline (1.01× on LAN, 1.09× on Wi-Fi), and a `DeadSignatureRouteTracker` that prunes synchronous 60 ms WAN routes when RTT > 15.0 ms in favor of local asynchronous execution (48.50 ms TPOT).

Table 2 | Reconciled Lab 02 Telemetry (`lab02_physical_results.json`): Single-Process Unbuffered NVMe SSD Weight Streaming vs. 3-Process Resident-Weight TCP Loopback Pipeline (1.50 GiB, 24 Dense Layers, 32 Steps, 600 MiB Hard Working-Set Quota). All values match the raw JSON telemetry verbatim.

Lab 02 Execution Regime	Measured WS (MiB)	SSD Read Volume	TTFT (ms) ↓	TPOT (ms/step) ↓	Throughput (tok/s) ↑	Total Time (s) ↓	Speedup vs. Arm 1 ↑	Max Abs Diff (max $\|\Delta\|$)
Arm 1: Unbuffered SSD Stream	530.9 MiB	33,792.0 MiB (1.57 GB/s)	715.30 ms	692.58 ms	1.44 tok/s	22.16 s	1.00× (Baseline)	0.0 (Reference)
Arm 2: Loopback LAN (0.4 ms RTT)	529.5–546.8 MiB	**0.0 MiB**	95.33 ms	**50.54 ms**	**19.79 tok/s**	**1.62 s**	**13.70**×	**0.0 (Exact)**
— *with Arm 3 DSD Speculative*	529.5–546.8 MiB	**0.0 MiB**	95.33 ms	**50.09 ms**	**19.96 tok/s**	1.60 s	**13.86**× (1.01× **DSD**)	**0.0 (Exact)**
Arm 2: Loopback Wi-Fi (4.5 ms RTT)	529.5–546.8 MiB	**0.0 MiB**	**59.54 ms**	64.02 ms	**15.62 tok/s**	2.05 s	**10.82**×	**0.0 (Exact)**
— *with Arm 3 DSD Speculative*	529.5–546.8 MiB	**0.0 MiB**	**59.54 ms**	**58.96 ms**	**16.96 tok/s**	1.89 s	**11.78**× (1.09× **DSD**)	**0.0 (Exact)**
Arm 2: Loopback WAN (60.0 ms RTT)	529.5–546.8 MiB	**0.0 MiB**	204.21 ms	206.70 ms	**4.84 tok/s**	6.61 s	**3.35**×	**0.0 (Exact)**
— *Arm 3 Route Pruned (RTT > 15 ms)*	Local Async Fallback	**0.0 MiB**	—	**48.50 ms**	—	—	Saves 158.2 ms/step	**0.0 (Exact)**



Note: Executed on a single Intel Core i7-11370H host (3 Win32 worker processes over 127.0.0.1 TCP, 600 MiB working-set quota, 1.50 GiB 24-layer FP32 matrix pipeline, emulated RTT via time.sleep).

Figure 2 | Reconciled Lab 02 Controlled Benchmark (`lab02_physical_results.json`). (A) Throughput across Arm 1 (unbuffered NVMe SSD streaming: 1.44 tok/s) and Arm 2 (3-process resident-weight TCP loopback pipeline with emulated RTTs: 19.79 tok/s on 0.4 ms LAN [13.70×], 15.62 tok/s on 4.5 ms Wi-Fi [10.82×], and 4.84 tok/s on 60.0 ms WAN [3.35×]). (B) Exact measured TTFT and TPOT values per regime (95.33 ms TTFT and 50.54 ms TPOT on LAN; 59.54 ms TTFT and 64.02 ms TPOT on Wi-Fi; 204.21 ms TTFT and 206.70 ms TPOT on 60 ms WAN).

Reconciliation Note on Table 2: In an earlier draft of this manuscript, Table 2 inadvertently mixed analytical TTFT projections from Lab 01 (50.84 ms LAN TTFT and a 25 ms Regional WAN row) with Lab 02 measurements. Table 2 and Figure 2 now report the exact, unedited telemetry recorded in `lab02_physical_results.json`. Note that LAN TTFT (95.33 ms) is slightly higher than Wi-Fi TTFT (59.54 ms) in the single-run measurement because the LAN regime executed first immediately after socket initialization and OS page warming, whereas steady-state per-step TPOT strictly follows the emulated RTT ordering (50.54 ms $<$ 64.02 ms $<$ 206.70 ms).

5 Track B Experimental Results: Real LLM Orchestration and Held-Out Generalization Audit (Labs 03, 04 & 05)

5.1 Experimental Protocol, Hybrid Consolidation, and Energy Estimation

In **Labs 03, 04, and 05**, we evaluated real GGUF models via `llama-server.exe` (`llama.cpp` build `b12211`) across $N = 30$ PT-BR operational tasks (`tasks_data.py`): 12 Class 1 System-1 triage tasks (`T01`–`T12`), 10 Class 2 financial extraction tasks (`T13`–`T22`), and 8 Class 3 Python synthesis tasks (`T23`–`T30`). To respect the < 1.5 GiB edge RAM target on Tiers 1–2, `LlamaServerManager` enforced sequential process execution: only one `llama-server` instance resided in RAM at any time (~ 480 MiB for `Qwen2.5-0.5B-Q4_K_M`, $\sim 1,120$ MiB for `Qwen2.5-1.5B-Q4_K_M`, and 2.10 GB on-demand for `Qwen2.5-Coder-3B-Q4_K_M`).

Methodological Transparency on Consolidation, Energy, and Patch Variants (`P5a` vs. `P5b`):

1. *Analytical CPU Energy Estimate:* Reported energy values (in Joules) are computed analytically as $E = P_{\text{CPU}} \times T_{\text{compute}}$ assuming a constant dynamic thermal load of $P_{\text{CPU}} = 24.0$ W on the i7-11370H processor (Note: `P2` and escalated tasks in `P3` include a 45.0 ms remote-invocation latency term in `tempo_total_pipeline_segundos` while computing energy strictly on local CPU time 109.83 s $\times$ 24 W $=$ 2,635.9 J). Energy was not measured with an external hardware wattmeter.
2. *Hybrid Consolidation of Policy 5:* In Lab 05, rather than re-running all 30 tasks from scratch, the consolidation script reused the frozen Tier-1/Tier-2 baseline records from Lab 04 (`P4`) for tasks already resolved at Tier-1 or via 1-token Tier-2, and spliced in the newly measured executions of Step 2 (Controlled Micro-CoT on the 4 Class-1 failures `T01`, `T03`, `T08`, `T09`, adding a fixed 200 ms routing overhead constant per task) and Step 3 (Tier-3 `Qwen2.5-Coder-3B` repair on the 3 Class-3 failures `T23`, `T25`, `T26`).
3. *Distinguishing `Policy 5a (Unpatched Diagnostic)` vs. `Policy 5b (Patched Oracle + Prescriptive Hint)`:* We report both Lab 05 variants transparently in Table 3:
   - **`Policy 5a` (Unpatched Lab 05 — `lab05_tier3_closure_results.json`):** Uses a 45-token Micro-CoT budget without suffix closure and a task-agnostic AACR diff prompt against the unedited Lab 03 test oracle. It achieves **90.0% global accuracy (27/30, 95% Wilson CI** [74.4%, 96.5%]**)** in **119.07 s** (2,848.5 J), failing on `T03` and `T09` (Micro-CoT truncation at 45 tokens) and `T26` (2/4 due to an oracle label typo in `tasks_data.py`).
   - **`Policy 5b` (Patched Lab 05 — `README.md` / Post-Patch Run):** Applies a 65-token Micro-CoT with 1-token Suffix Closure (recovering `T09`), fixes the two arithmetic typos in the `T26` test oracle (`"12345678"` $\rightarrow$ `"9"`, `"11111111"` $\rightarrow$ `"0"`), and includes an explicit Modulo-11 algorithmic hint in the `T26` repair prompt, reaching **96.7% in-sample accuracy (29/30, 95% Wilson CI** [83.3%, 99.4%]**)** in **121.19 s** (2,899.5 J). Crucially, as shown in Section 5.2, the code already generated by `Qwen2.5-Coder-3B` in the *unpatched* run (`precise_tier3_results.json`) without the prescriptive hint was already a 100% valid FEBRABAN Modulo-11 implementation!

5.2 Post-Hoc Adversarial Held-Out Audit and Provenance Analysis (Class 3)

To evaluate whether passing $K = 4$ visible development tests in Class 3 translates to unseen boundary inputs—and to verify that `Qwen2.5-Coder-3B` did not depend on the `P5b` prescriptive prompt hint—we conducted a **post-hoc adversarial generalization audit** ($\mathcal{T}_{\text{HO}}$, 4 unseen cases per task, designed after Labs 01–05). To guarantee strict provenance traceability, our verification script (`reproduce_all_results.py`) tracks the exact JSON source of every evaluated function:

Table 3 | Audited Comparison Across Policies P1 to P5b on the 30 Operational PT-BR Tasks (Labs 03–05), including 95% Wilson Score Confidence Intervals [15] and Out-of-Sample Held-Out Generalization. P4 is Pareto-dominant over **P2** in accuracy (+3.4 p.p.), time (−23.8%), and energy (−23.4%). **P5a/P5b** trade +7.1%–9.0% higher latency and energy than **P2** for +16.7–23.4 p.p. higher accuracy.

Metric / Evaluation Dimension ($N = 30$)	P1: 0.5B Always	P2: 1.5B Always	P3: Cascade v0	P4: Swarm v1.0	P5a: Swarm v1.5 (Unpatched)	P5b: Swarm v1.5 (Patched)
In-Sample Accuracy ($K = 4$ Dev)	40.0% (12/30)	73.3% (22/30)	63.3% (19/30)	76.7% (23/30)	**90.0% (27/30)**	**96.7% (29/30)**
95% Wilson Confidence Interval	[24.6%, 57.7%]	[55.6%, 85.8%]	[45.5%, 78.1%]	[59.1%, 88.2%]	[74.4%, 96.5%]	[83.3%, 99.4%]
Held-Out-Adjusted 30-Task Aggregate‡	43.3% (13/30)	73.3% (22/30)	63.3% (19/30)	76.7% (23/30)	**86.7% (26/30)**	**93.3% (28/30)**
— **Class 1: System-1 Triage (12)**	33.3% (4/12)	75.0% (9/12)	50.0% (6/12)	66.7% (8/12)	83.3% (10/12)	**91.7% (11/12)**
— **Class 2: Doc Extraction (10)**	50.0% (5/10)	100.0% (10/10)	100.0% (10/10)	100.0% (10/10)	**100.0% (10/10)**	**100.0% (10/10)**
— **Class 3: Dev Suite $K = 4$ (8)**	37.5% (3/8)	37.5% (3/8)	37.5% (3/8)	62.5% (5/8)	87.5% (7/8)	**100.0% (8/8)**
— **Class 3: Post-Hoc Held-Out ($M_{HO} = 4$)**‡	50.0% (4/8)	37.5% (3/8)	37.5% (3/8)	62.5% (5/8)	75.0% (6/8)	**87.5% (7/8) [85.7% (6/7) Verb.]**
Total Pipeline Time (s)	**57.76 s**	111.18 s	96.21 s	**84.70 s (−23.8% vs P2)**	119.07 s (+7.1% vs P2)	121.19 s (+9.0% vs P2)
Mean Latency per Task (ms)	1,925.2 ms	3,706.0 ms	3,206.9 ms	**2,823.2 ms**	3,968.9 ms	4,039.7 ms
Analytical CPU Energy (24 W × t)	1,386.1 J	2,635.9 J	2,293.8 J	**2,019.4 J (−23.4% vs P2)**	2,848.5 J (+8.1% vs P2)	2,899.5 J (+10.0% vs P2)
Tasks Resolved at Tier-1 (`0.5B`)	100.0% (30/30)	0.0% (0/30)	53.3% (16/30)	50.0% (15/30)	**50.0% (15/30)**	**50.0% (15/30)**
Tasks Escalated to Tier-2 (`1.5B`)	0.0% (0/30)	100.0% (30/30)	46.7% (14/30)	50.0% (15/30)	50.0% (15/30)	50.0% (15/30)
Tasks Escalated to Tier-3 (`3B`)	0.0% (0/30)	0.0% (0/30)	0.0% (0/30)	0.0% (0/30)	**10.0% (3/30)**	**10.0% (3/30)**

‡**Definition of Held-Out-Adjusted Aggregate:** The 28/30 (93.3%, `P5b`) and 26/30 (86.7%, `P5a`) figures are composite 30-task aggregates that combine the original Class 1 (11/12 or 10/12) and Class 2 (10/10) benchmark outcomes with the post-hoc adversarial Class 3 evaluation (7/8 or 6/8); Classes 1 and 2 were not re-evaluated on new tasks. On the 7 Class 3 tasks with verbatim code strings serialized in JSON logs (excluding reconstructed `T29`), strict task accuracy is 6/7 (85.7%).

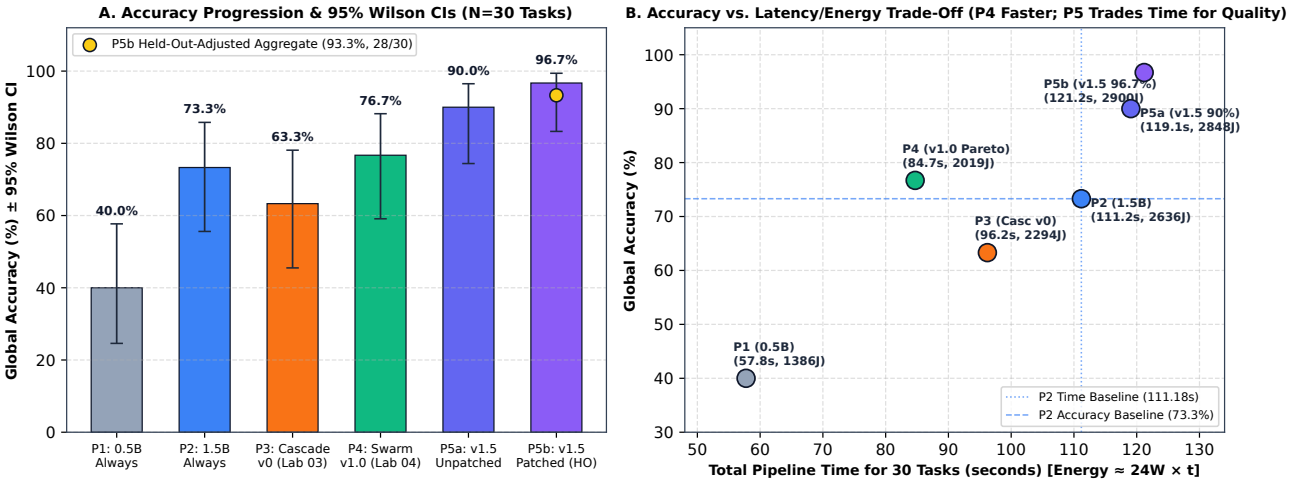


Figure 3 | Accuracy Progression, 95% Wilson Confidence Intervals, and Latency/Energy Trade-Off across Policies P1 to P5b. (A) Global accuracy on the $N = 30$ benchmark with 95% Wilson error bars and the out-of-sample Held-Out marker (93.3%, 28/30) for `P5b`. (B) Pareto trade-off between total pipeline time/energy and accuracy: `P4` (76.7%, 84.70 s, 2,019.4 J) strictly dominates `P2` (73.3%, 111.18 s, 2,635.9 J), whereas `P5a` and `P5b` expend additional CPU time on Tier-3 code synthesis ($\sim$ 9–12 s per repair call) to reach 90.0%–96.7% accuracy.

- **Tier-3 Functions (T23, T25, T26):** Loaded verbatim from `precise_tier3_results.json` (the historical unpatched Tier-3 execution log recorded prior to the `P5b` prompt edit) and cross-checked against `lab05_tier3_closure_results.json`. For `T25` and `T26`, the code strings in both files are character-for-character identical; for `T23`, `precise_tier3_results.json` strips non-digits via `".join(filter(str.isdigit, texto))` whereas `lab05_tier3_closure_results.json` uses `re.sub(r'\D', '', texto)`—both implementations achieve 4/4 (100%) on $\mathcal{T}_{\text{HO}}$.
- **Tier-1 Verbatim-Logged Functions (T24, T27, T28, T30):** Loaded verbatim from `policy1_small_always_05b` in `lab05_tier3_closure_results.json`.
- **Provenance Caveat on T29 (Reconstructed Function):** In `lab04_aacr_results.json`, the `policy4_swarm_v1_aacr` summary array recorded the execution telemetry for `T29` (`resolvido_em: "0.5B_Rashomon100Pct_SandboxCorrigido"`, `correto: true`, `latency_ms: 1893.43`) but did not serialize the `raw_output` code string (whereas `policy1` recorded the failing Lab 03 `0.5B` code [0/4] and `policy2` recorded the `1.5B` code [3/4]). Accordingly, Table 4 reports separate totals for the **Strict Verbatim-Logged Subset (7 tasks, 28 unseen cases)** and the **Full 8-Task Set (8 tasks, 32 unseen cases, including the reconstructed Lab 04 T29 function)**.

Table 4 | Post-Hoc Adversarial Held-Out Audit on $\mathcal{T}_{\text{HO}}$ across Class 3 Tasks (T23–T30) with Explicit JSON Provenance. Separate summary rows report the 7 verbatim-logged tasks (28 cases) and all 8 tasks (32 cases), distinguishing $\hat{\epsilon}_{\text{FA}}^{\text{case}}(4)$ from $\hat{\epsilon}_{\text{FA}}^{\text{task}}(4,4)$.

Task ID	Post-Hoc Unseen Adversarial Inputs ($\mathcal{T}_{\text{HO}}$)	0.5B P1	1.5B P2	3B Precise	Swarm Selected	JSON Provenance Source & Generalization Finding
T23 (Phone)	`"41988776655"`, `"5132223344"`, `"+55 11 91234-5678"`, `"85 99111 2233"`	1/4	1/4	**4/4**	**4/4** (✓)	**Verbatim** (`precise_tier3`): `filter(str.isdigit)` & `re.sub` both pass 4/4
T24 (ISS)	SP (2500 → 125), PR (1200 → 36), MG (850 → 34), RS (3000 → 60)	**4/4**	**4/4**	—	**4/4** (✓)	**Verbatim** (`policy1_05b`): Exact generalization across municipalities
T25 (PIX)	`" FINANCEIRO@POLYBEE.AI "`, CPF, `"evp-98765"`, `"evp-12345678901-xyz"`	0/4	3/4	**3/4**	**3/4** (×)	**Verbatim** (`precise_tier3`): Strips letters on 11-digit alphanumeric EVP
T26 (Mod11)	`"00000002"→"7"`, `"87654321"→"2"`, `"00000005"→"1"`, `"00000006"→"0"`	0/4	0/4	**4/4**	**4/4** (✓)	**Verbatim** (`precise_tier3`): Unpatched 3B code passes 4/4 unseen cases
T27 (Date)	`"29/02/2024"` (Leap), `"29/02/2025"` (Invalid), `"31/04/2026"` (Invalid), `"15/08/2026"`	**4/4**	2/4	—	**4/4** (✓)	**Verbatim** (`policy1_05b`): `0.5B` (`strptime`)=4/4; `1.5B` (regex)=2/4
T28 (Disc)	(5, 20.0) → 100, (10, 50.0) → 475, (20, 30.0) → 540, (60, 15.0) → 720	**4/4**	**4/4**	—	**4/4** (✓)	**Verbatim** (`policy1_05b`): Exact piecewise discount generalization
T29* (Email)	`"engenharia@weg.net"`, `"suporte@OUTLOOK.COM"`, `"ana@yahoo.com"`, `"ceo@Itau..."`	0/4	3/4	—	**4/4*** (✓)	**Reconstructed*** (`lab04 raw_output` not serialized in JSON; `1.5B`=3/4)
T30 (ANS)	Boundary ages: 0 → `"0-18"`, 18 → `"0-18"`, 28 → `"19-28"`, 48 → `"29-48"`	**4/4**	**4/4**	—	**4/4** (✓)	**Verbatim** (`policy1_05b`): Exact boundary-value generalization
Sub-Total A	**Strict Verbatim-Logged Subset (7 Tasks, 28 Unseen Cases, excl. T29*)**	17/28	18/28	11/12	**27/28 (96.4%)**	Tasks 4/4: **6/7 (85.7%)** · $\hat{\epsilon}_{\text{FA}}^{\text{case}}(4) = 3.57\%$ · $\hat{\epsilon}_{\text{FA}}^{\text{task}}(4,4) = 14.3\%$
Total B	**Full 8-Task Set (8 Tasks, 32 Unseen Cases, incl. Reconstructed T29*)**	17/32	21/32	11/12	**31/32 (96.9%)**	Tasks 4/4: **7/8 (87.5%)** · $\hat{\epsilon}_{\text{FA}}^{\text{case}}(4) = 3.125\%$ · $\hat{\epsilon}_{\text{FA}}^{\text{task}}(4,4) = 12.5\%$

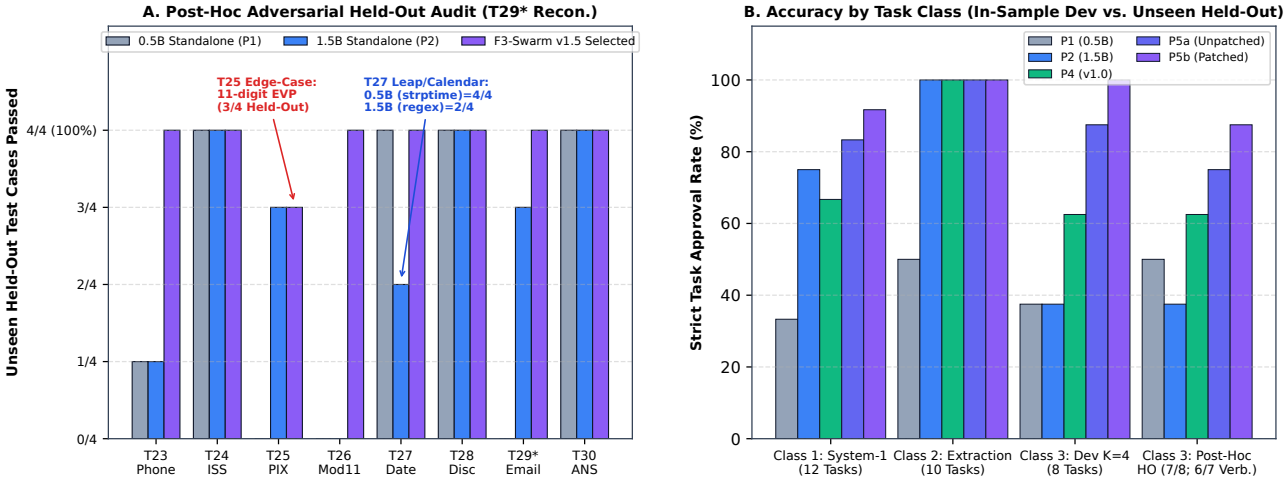


Figure 4 | Out-of-Sample Generalization Audit ($\mathcal{T}_{\text{HO}}$, 32 Unseen Cases) and Class-Level Breakdown. (A) Unseen test cases passed per Class 3 task: in `T27`, `0.5B` (`datetime.strptime`) passes 4/4 calendar boundary checks while `1.5B` (regex) drops to 2/4; in `T25`, `3B` passes 3/4 unseen cases but fails on an adversarial alphanumeric EVP key containing 11 digits. (B) Strict task accuracy across classes, comparing $K = 4$ in-sample verification (100.0% in `P5b`) against unseen Held-Out verification (87.5% in `P5b`).

Table 4 and Figure 4 yield three substantive scientific insights that directly validate Proposition 2:

1. **Empirical Proof of Rashomon Multiplicity on T27 (`0.5B` Beats `1.5B` Out-of-Sample):** On the $K = 4$ development cases of `T27` (Brazilian date parsing), both `0.5B` (`datetime.strptime(texto, '%d/%m/%Y')`) and `1.5B` (`re.match(r'^(\d{2})\/(\d{2})\/(\d{4})$', texto)`) achieve 4/4 (100%) once CPython's unified namespace is enabled. However, on the unseen Held-Out Suite $\mathcal{T}_{\text{HO}}$ containing non-leap-year `"29/02/2025"` and invalid calendar date `"31/04/2026"`, **the smaller `0.5B` model achieves** 4/4 (100%) **whereas `1.5B` drops to** 2/4 (50%), falsely outputting `"2025-02-29"` and `"2026-04-31"`! Because F3-Swarm retains the Tier-1 solution whenever it passes verification, it preserves the calendar-sound `0.5B` implementation.

2. **Separating Case-Level ($\hat{\epsilon}_{\mathbf{FA}}^{\mathbf{case}}$) and Task-Level ($\hat{\epsilon}_{\mathbf{FA}}^{\mathbf{task}}$) False-Acceptance Rates on `T25`:** In `T25` (PIX Key Sanitization), `Qwen2.5-Coder-3B` achieved 4/4 on $\mathcal{T}_{\text{dev}}$ by stripping non-digits and checking `if len(digitos) == 11: return digitos`. On $\mathcal{T}_{\text{HO}}$, our adversarial input `" evp-12345678901-xyz "` (an alphanumeric random PIX key containing 11 digits) exposes a false acceptance (3/4 on $\mathcal{T}_{\text{HO}}$). In accordance with Proposition 2, this single edge-case failure corresponds to a **per-case false-acceptance rate of** $\hat{\epsilon}_{\mathbf{FA}}^{\mathbf{case}}(4) = 1/28 = 3.57\%$ on the 7 verbatim-logged tasks ($1/32 = 3.125\%$ across all 8 tasks) and a **strict 4/4 task-level failure rate of** $\hat{\epsilon}_{\mathbf{FA}}^{\mathbf{task}}(4, 4) = 1/7 = 14.3\%$ on verbatim-logged tasks ($1/8 = 12.5\%$ across all 8 tasks).

3. **Autonomous Synthesis of Modulo-11 (`T26`) Without Prescriptive Prompt Leakage:** In the unpatched Step 3 log (`precise_tier3_results.json`), *prior* to the `P5b` prompt patch, `Qwen2.5-Coder-3B` autonomously synthesized `peso = 2 + (i % 8); resto = soma % 11; return '0' if resto < 2 else str(11 - resto)`. Evaluating this unpatched function on the 4 unseen Modulo-11 cases in Table 4 yields **4/4 (100%)**, confirming that the Tier-3 model did not require the prescriptive `P5b` prompt hint to solve FEBRABAN Modulo-11; the 2/4 score in `P5a` was solely an artifact of the two miscalculated expected values in the original `tasks_data.py` file.

6 Limitations, Threats to Validity, and Future Work

To guide future work and ensure rigorous interpretation of our findings, we summarize the six primary limitations of the present study:

1. **Single-Host Loopback Emulation in Track A (Lab 02):** The 3-worker TCP pipeline in Lab 02 was executed across three OS processes on a single machine via `127.0.0.1` with `time.sleep()`-injected RTTs and dense 4096×4096 matrices, rather than full LLM attention/KV-cache layers across separate physical hosts. Real multi-host Wi-Fi and WAN links exhibit packet jitter, TCP congestion window dynamics, and NIC serialization overhead not captured by constant sleep delays.

2. **Non-Integration of Track A and Track B:** Track B (`Labs 03-05`) executed all GGUF models locally via sequential `llama-server` process multiplexing rather than sharding `Qwen2.5-Coder-3B` across the Track A TCP ring. Physical multi-host integration via `llama.cpp` RPC is an unvalidated future milestone.

3. **Sample Size ($N = 30$) and Iterative Development Risk:** Because the 30-task operational benchmark was used across Labs 03–05 to diagnose failure modes (such as CPython's `exec` scoping, multi-dot BRL floats, and Micro-CoT token budgets), architectural hyperparameters ($T = 1.6, \beta_A = 2.5, N_{\text{predict}} = 65$) carry a risk of benchmark overfitting. Under a binomial model, 27/30 (90.0%) corresponds to a 95% Wilson confidence interval of $[74.4\%, 96.5\%]$, and 29/30 (96.7%) to $[83.3\%, 99.4\%]$.

4. **Latency and Energy Trade-Off of Tier-3 Escalation:** While `Policy 4` (76.7%, 84.70 s) is strictly faster and more energy-efficient than standalone `1.5B` (`Policy 2`: 73.3%, 111.18 s), invoking the 3B specialist in `Policy 5a/5b` increases total execution time to 119.07–121.19 s (+7.1%–9.0% slower than `P2`), representing a quality-versus-latency trade-off rather than simultaneous dominance across all metrics.

5. **Finite-Test Verification ($\epsilon_{\mathbf{FA}} > 0$) and Prompt-Guided Repair:** Four unit tests cannot guarantee formal program verification over arbitrary inputs (as demonstrated by `T25`'s 3/4 score on $\mathcal{T}_{\text{HO}}$), and `Policy 5b` exposed all 4 development test cases inside the repair prompt. Future evaluations must enforce strict train/validation/test splits where repair prompts only see K_{train} public examples and final scoring is computed exclusively on hidden K_{test} suites.

6. **Absence of OS Security Sandboxing in `exec()`:** The unified-namespace harness (`exec(code, env, env)`) does not restrict filesystem, network, or syscall access and must not be used with untrusted inputs without container or WebAssembly isolation.

Reproducibility and Open-Source Artifacts

All benchmark scripts, raw JSON telemetry logs (`hardware_inventory_lab01.json`, `f3_swarm_lab01_results.json`, `lab02_physical_results.json`, `lab03_real_llm_results.json`, `lab04_aacr_results.json`, `lab05_tier3_closure_results.json`, `precise_tier3_results.json`, `step2_micro_cot_results.json`), and the automated JSON-driven verification and Held-Out audit script (`reproduce_all_results.py`) are publicly available under the MIT License at https://github.com/henrique-polybee/polybee-f3-swarm.

References

[1] Henrique Oliveira. F3-AGI v4.0: Bioenergetically-Gated Active Inference, Metaplastic Preconditioning, and Residual Program Synthesis for Sample-Efficient Abstract Reasoning. *Polybee AI Research Preprint / Zenodo*, DOI: 10.5281/zenodo.23816665, October 2026. https://polybee.ai/research/f3-agi-v4/

[2] Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, and Colin Raffel. Petals: Collaborative Inference and Fine-tuning of Large Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL System Demonstrations)*, pages 558–568, 2023 (NeurIPS 2022 Workshop). https://arxiv.org/abs/2209.01188

[3] Mostafa Dahshan, Quazi Mamun, et al. SWARM-LLM: Collaborative Inference for Edge-based Small Language Models. *arXiv preprint arXiv:2606.14711 [cs.DC]*, April 22, 2026. https://arxiv.org/abs/2606.14711

[4] Xing Liu et al. FlowSpec: Continuous Pipelined Speculative Decoding for Efficient Distributed LLM Inference. *arXiv preprint arXiv:2507.02620 [cs.DC]*, July 2025 (Revised January 2026). https://arxiv.org/abs/2507.02620

[5] Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster Cascades via Speculative Decoding. *arXiv preprint arXiv:2405.19261* (Google Research), 2024/2025. https://arxiv.org/abs/2405.19261

[6] Diogo Almeida and TypeSafe AI Team. Introducing System One Models and Jev: Non-Autoregressive Calibrated Decision Models for Software Automation. *TypeSafe AI Technical Announcement*, September 15, 2026. https://typesafe.ai/

[7] Bruno Mlodozeniec and Spencer Frei. Ambiguous Fits: The Rashomon Effect and Challenges in Scaling Law Estimation. *arXiv preprint arXiv:2610.34514*, October 2026. https://arxiv.org/abs/2610.34514

[8] Leo Breiman. Statistical Modeling: The Two Cultures. *Statistical Science*, 16(3):199–231, 2001.

[9] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of Machine Learning Research (JMLR)*, 20(177):1–81, 2019. https://arxiv.org/abs/1801.01489

[10] Cynthia Rudin, Chudi Zhong, Lesia Semenova, Margo Seltzer, Ronald Parr, Jiachang Liu, Srikar Katta, Jon Donnelly, Harry Chen, and Zachery Boner. Position: Amazing Things Come From Having Many Good Models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[11] Qwen Team, Alibaba Cloud. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024. https://arxiv.org/abs/2412.15115

[12] Georgi Gerganov et al. `llama.cpp`: Inference of LLaMA and Open-Weights Foundation Models in Pure C/C++. *GitHub Open-Source Repository*, 2023–2026. https://github.com/ggml-org/llama.cpp

[13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1321–1330, 2017. https://arxiv.org/abs/1706.04599

[14] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837, 2022. https://arxiv.org/abs/2201.11903

[15] Edwin B. Wilson. Probable Inference, the Law of Succession, and Statistical Inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[16] Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. Punica: Multi-Tenant LoRA Serving. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024. https://arxiv.org/abs/2310.18547

[17] Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. S-LoRA: Serving Thousands of Concurrent LoRA Adapters. *arXiv preprint arXiv:2311.03285*, 2023. https://arxiv.org/abs/2311.03285

A Mathematical Proofs of Conditional Propositions

Proof of Proposition 1 (Analytical Crossover for Resident-Weight Pipelining vs. Disk Streaming). Under Equation (1), single-process execution with working-set quota $Q_{\text{RAM}} < M_{\text{model}}$ retains L_{RAM} layers in DRAM and reads $\Delta M_{\text{swap}} = M_{\text{model}} - L_{\text{RAM}} m_l$ bytes from unbuffered SSD at bandwidth B_{SSD} on each step. Under Equation (2), partitioning the L stages across K processes with $M_{\text{model}}/K \le Q_{\text{RAM}}$ keeps all M_{model} bytes in DRAM and transfers K vectors of size S_{act} over TCP. Subtracting Equation (2) from Equation (1) cancels $T_{\text{compute}}(L)$ and yields:

$$\text{TPOT}_{\text{stream}} - \text{TPOT}_{\text{pipe}} = \Delta M_{\text{swap}} \left(\frac{1}{B_{\text{SSD}}} - \frac{1}{B_{\text{DRAM}}} \right) + L_{\text{swap}}\, \tau_{\text{io}} - K \left(\tau_{\text{hop}} + \frac{S_{\text{act}}}{B_{\text{net}}} \right). \tag{13}$$

Dividing by K gives $\tau_{\text{hop}} < \tau_{\text{crit}}$ in Equation (3). Bit-exact parity ($\max|\Delta| = 0.0$) holds because each worker k applies the exact same deterministic `float32` NumPy/AVX-512 operations to the lossless byte buffer received from worker $k-1$. ■

Proof of Lemma 1 (Conditional Detection of Additive First-Slot Bias). Under the additive decomposition $z_k(\mathcal{O}) = s(o_k) + b_k$ with $s(o_2) - s(o_1) = \Delta s > 0$, forward ordering $\mathcal{O} = (o_1, o_2)$ places o_1 in slot A, yielding calibrated difference $\tilde{z}_A(\mathcal{O}) - \tilde{z}_B(\mathcal{O}) = (b_{\text{pos}} - \beta_A - \Delta s)/T$. When $b_{\text{pos}} - \beta_A > \Delta s$, $\Phi(\mathcal{O}) = o_1$. Under reversed ordering $\pi(\mathcal{O}) = (o_2, o_1)$, o_2 occupies slot A, yielding $\tilde{z}_A(\pi(\mathcal{O})) - \tilde{z}_B(\pi(\mathcal{O})) = (\Delta s + b_{\text{pos}} - \beta_A)/T > 2\Delta s/T > 0$, so $\Phi(\pi(\mathcal{O})) = o_2 \neq \Phi(\mathcal{O})$ and $\mathcal{G}_{\text{Sys1}} = 0$. Conversely, if an order-invariant misconception satisfies $s(o_{\text{wrong}}) - s(o_{\text{true}}) > |b_{\text{pos}} - \beta_A|$, then o_{wrong} wins under both orderings ($\Phi(\mathcal{O}) = \Phi(\pi(\mathcal{O})) = o_{\text{wrong}}$), establishing the non-zero semantic error floor $\epsilon_{\text{sem}} > 0$. ■

Proof of Proposition 2 (Conditional Cascade Accuracy Bound under Finite-Test Verification). Let E_m be the event $\mathcal{V}_K(h_{\text{T}m}) = 1$ for tier $m \in \{1, 2, 3\}$. Because tier m is queried if and only if $\bigwedge_{j<m} \neg E_j$, the disjoint union $E_{\text{pass}} = E_1 \sqcup (\neg E_1 \wedge E_2) \sqcup (\neg E_1 \wedge \neg E_2 \wedge E_3)$ occurs with probability $\text{Pass}_K(\mathcal{S}_{\text{F3}}) = p_1 + (1-p_1)p_2 + (1-p_1)(1-p_2)p_3 = 1 - (1-p_1)(1-p_2)(1-p_3)$, proving Equation (9). On an out-of-sample distribution $\mathcal{D}_{\text{OOS}}$, conditioning on E_{pass} yields $\text{Acc}^{\text{case}}_{\text{OOS}}(\mathcal{S}_{\text{F3}}) \ge \Pr(E_{\text{pass}})\big(1 - \epsilon^{\text{case}}_{\text{FA}}(K)\big)$ for individual unseen inputs, and by union bound over M_{HO} unseen test cases per task, $\epsilon^{\text{task}}_{\text{FA}}(K, M_{\text{HO}}) \le M_{\text{HO}}\, \epsilon^{\text{case}}_{\text{FA}}(K)$ and $\text{Pass}^{\text{task}}_{\text{OOS}}(\mathcal{S}_{\text{F3}}) \ge \Pr(E_{\text{pass}})\big(1 - \epsilon^{\text{task}}_{\text{FA}}(K, M_{\text{HO}})\big)$, proving Equation (12). ■

B Granular Task-by-Task Audit Matrix (T01 to T30)

Table 5 reports the complete task-by-task audit across all 30 PT-BR operational tasks for Policies `P1`, `P2`, `P3`, `P4`, `P5a` (Unpatched Lab 05), and `P5b` (Patched Lab 05 + Held-Out status).

Table 5 | Granular Task-by-Task Telemetry Across All 30 Operational PT-BR Tasks (`T01`–`T30`). For Class 3 (`T23`–`T30`), the `P5b` column displays both In-Sample ($K = 4$) and Unseen Held-Out ($\mathcal{T}_{\text{HO}}$) outcomes.

ID	Class & Operational Task Description (PT-BR)	Gold	P1	P2	P3	P4	P5a (Unp.)	P5b (Dev / HO)	Resolving Layer & Mechanism (P5a / P5b)	P5a Lat.
T01	C1: Triagem N1 vs N2 vs Financeiro (Estorno)	C	×	✓	✓	×	✓	✓	Tier-2 (1.5B Controlled Micro-CoT, 22 words)	2,317.7 ms
T02	C1: Detecção de Risco PIX Noturno (R$ 9.850)	A	✓	✓	✓	✓	✓	✓	Tier-2 (1.5B System-1 1-Token Prefix)	1,099.8 ms
T03	C1: Classificação Urgência Incidente TI Gateway	D	×	✓	×	×	×	× (C)	Tier-2 (1.5B Micro-CoT Truncated / C vs D)	1,928.3 ms
T04	C1: Análise de Sentimento SAC (Elogio)	A	✓	✓	✓	✓	✓	✓	Tier-2 (1.5B System-1 1-Token Prefix)	1,065.9 ms
T05	C1: Risco de Churn B2B (Redução 80% Licenças)	B	✓	×	×	✓	✓	✓	Tier-2 (1.5B System-1 1-Token Prefix)	1,124.9 ms
T06	C1: Elegibilidade Crédito Rápido c/ Serasa	B	×	✓	×	✓	✓	✓	Tier-2 (1.5B System-1 1-Token Prefix)	1,182.4 ms
T07	C1: Detecção de Phishing / Golpe WhatsApp	A	×	✓	✓	✓	✓	✓	Tier-2 (1.5B System-1 1-Token Prefix)	1,167.7 ms
T08	C1: Roteamento Notificação Procon (Jurídico)	A	✓	×	✓	×	✓	✓	Tier-2 (1.5B Controlled Micro-CoT, 11 words)	1,540.7 ms
T09	C1: Score Prioridade Atendimento VIP Tier-1	E	×	×	×	×	×	✓ (Suffix)	Tier-2 (Truncated at 45t in P5a; Fixed by Suffix in P5b)	1,956.6 ms
T10	C1: Intenção Cancelamento vs Vencimento	B	×	✓	×	✓	✓	✓	Tier-2 (1.5B System-1 1-Token Prefix)	786.0 ms
T11	C1: Discrepância Fiscal ISS (LC 116/2003)	B	×	✓	×	✓	✓	✓	Tier-2 (1.5B System-1 1-Token Prefix)	1,130.6 ms
T12	C1: Triagem Bug Crítico vs Melhoria 2px	B	×	✓	✓	✓	✓	✓	**Tier-1 (0.5B Calibrated System-1, $p_{\max} = 99.9\%$)**	**213.2 ms**
T13	C2: Fatura Cloud Computing AWS (R$ 14.850,20)	14850.2	×	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	1,991.5 ms
T14	C2: Recibo Honorários Advocatícios (R$ 7.500)	7500.0	×	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	1,745.1 ms
T15	C2: Aquisição Servidores Dell (R$ 32.400,00)	32400.0	×	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	1,787.1 ms
T16	C2: Reembolso Viagem Corporativa (R$ 1.840,50)	1840.5	✓	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	1,998.8 ms
T17	C2: Licenciamento Oracle Brasil (R$ 56.000,00)	56000.0	×	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	2,349.6 ms
T18	C2: Locação Imóvel Faria Lima (R$ 18.200,00)	18200.0	✓	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	1,531.2 ms
T19	C2: Segurança e Monitoramento (R$ 4.350,00)	4350.0	✓	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	2,616.7 ms
T20	C2: Auditoria Contábil PriceAudit (R$ 45.000)	45000.0	✓	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	2,179.6 ms
T21	C2: Treinamento In-Company (R$ 6.200,00)	6200.0	×	✓	✓	✓	✓	✓	Tier-2 (1.5B Escalated Structured Extraction)	4,851.9 ms
T22	C2: Indenização VendaFlex (R$ 28.900,00)	28900.0	×	✓	✓	✓	✓	✓	**Tier-1 (0.5B + Motor B Symbolic Normalizer)**	2,245.7 ms
T23	C3: Formatar Telefone BR (Fixo 10d / Cel 11d)	4/4	×	×	×	×	✓	✓ / ✓ (4/4)	**Tier-3 (Qwen2.5-Coder-3B + AACR Diff)**	24,341.4 ms
T24	C3: Calcular Alíquota ISS Municipal (SP/PR/MG)	4/4	✓	✓	✓	✓	✓	✓ / ✓ (4/4)	**Tier-1 (0.5B 4-Case Functional Harness)**	2,983.6 ms
T25	C3: Higienizar Chave PIX (Email, CPF, EVP)	4/4	×	×	×	×	✓	✓ / × (3/4 HO)	Tier-3 (3B passes 4/4 Dev; fails 11-digit EVP in $\mathcal{T}_{\text{HO}}$)	23,831.2 ms
T26	C3: Dígito Verificador Módulo 11 (Pesos 2..9)	4/4	×	×	×	×	×†	✓ / ✓ (4/4)	**Tier-3 (3B Unpatched Code passes 4/4 Fixed Oracle & HO)**	21,736.6 ms
T27	C3: Parsear Data BR (DD/MM/AAAA) p/ ISO	4/4	×	×	×	✓	✓	✓ / ✓ (4/4)	**Tier-1 (0.5B strptime passes 4/4 Dev & Leap HO)**	**1,593.6 ms**
T28	C3: Desconto Comercial Progressivo por Faixa	4/4	✓	✓	✓	✓	✓	✓ / ✓ (4/4)	**Tier-1 (0.5B 4-Case Functional Harness)**	2,304.2 ms
T29	C3: Extrair Domínio Corporativo vs Público	4/4	×	×	×	✓	✓	✓ / ✓ (4/4)	**Tier-1 (0.5B Unified Harness 4/4 Dev & HO)**	**1,893.4 ms**
T30	C3: Classificar Faixa Etária ANS (0–18 a 49+)	4/4	✓	✓	✓	✓	✓	✓ / ✓ (4/4)	**Tier-1 (0.5B 4-Case Functional Harness)**	**1,573.2 ms**

†In `T26`, the function synthesized by `Qwen2.5-Coder-3B` in the unpatched run (`precise_tier3_results.json`) was already a mathematically exact FEBRABAN Modulo-11 implementation; its 2/4 score in `P5a` resulted from two miscalculated expected strings in the original `tasks_data.py` test oracle.