Satiation Benchmark: Reference Harness, Adapters, and Telemetry Attribution

David Charlot, Open Interface Engineering

Research study · draft · Open Interface Engineering · research.openie.dev

Revision `1561a2a` (22 November 2026) · Web: research.openie.dev/papers/satb-harness/ · Archival LaTeX: /pdfs/latex/satb-harness.pdf (.tex)

Measurement. Research study, not a journal final. Verified across `5 / 5` **silicon and OS archetypes**: physical Texas Instruments `INA238` (`VIN L1`) and three-channel `INA3221` (`VDD_GPU`, `VDD_CPU_SOC_MSS`, `VIN_SYS_5V0 L2`) shunt `measured_j` on `NVIDIA Jetson AGX Thor` (`aarch64` Tegra Linux, shared host; `A2` net placement attribution withheld per `Falsifier 6`), plus rootless on-chip `reported_j` on `Apple M5 Max` (`macOS 27.2.1 IOReport L2 + proc_pid_rusage L3`), `Framework AMD Ryzen AI Max+ 395` Strix Halo (`0440 root:energy PLATYPUS`-isolated `RAPL L2 + Radeon 8060S PPT`), `Intel Core i9-13900H` Raptor Lake (`RAPL psys L1 + package-0 L2`), and `Intel Core i7-1165G7 + NVIDIA RTX 2060` on `Windows 11 Pro` (build `26300`, rootless `0x20000 nvml.dll` cumulative `mJ L2` + Win32 `L3`; user-mode `CPU RAPL = not_available`). Closed-API server joules remain `not_available (N)`. No fabricated citations.

A fixed sequential Rust reference harness (Lane B), a single proxy-enforced adapter bridge (Shapes D, M, H, and I), a twelve-check conformance suite (C1 to C12), and a five-level rootless telemetry attribution engine (L0 to L4 crossed with labels A1 to N). Verified empirically across five hardware archetypes: Apple M5 Max (arm64 macOS IOReport and proc_pid_rusage reported_j), Tegra aarch64 Jetson Thor (physical whole-board and three-channel rail-separated current-shunt measured_j), Framework Desktop Strix Halo (x86-64 Linux 0440 root:energy group-isolated package-energy reported_j plus graphics-package power tracking), Intel Core i9-13900H Raptor Lake (x86-64 Linux platform L1 and package-0 L2 reported_j), and Intel Core i7-1165G7 Tiger Lake plus GeForce 2060 on Windows 11 (x86-64 build 26300 rootless cumulative discrete-graphics L2 reported_j, Win32 Job Object and cycle accounting, and processor-package Label N). Raw measured_j, reported_j, and L4 logical channels stay strictly separated and are never summed; because co-tenant workloads ran on the shared hosts (dedicated_host is false), A2 net placement attribution is withheld per Falsifier 6.

Contents

0. Overview **2**

1. Terms, acronyms, and attribution labels **3**
1.1 Lineage, design invariants, and same-house disclosure . . . 4

2. Survey of Agent Harnesses and Why Context Policy Is Part of the System **4**

3. Reference Harness (`Lane B`), `ContextRule_v0`, and `SATB-IO` Protocol **6**
3.1 Sequential execution loop and fixed-versus-configurable split . . . 6
3.2 Canonical four-way outcome mapping (`OQ-3`) . . . 7

4. Adapter Bridge (`Shapes D, M, H, I`), Five-Block `SUT` Fingerprint, and `C1-C12` Conformance **7**
4.1 Four adapter shapes behind one supervisor and proxy . . . 7
4.2 Five-block `SUT` fingerprint and session validity . . . 7
4.3 Mandatory `C1-C12` adapter conformance suite . . . 8

5. Five-Level Telemetry Attribution (`L0-L4 × A1-N`), Platform Channel Maps, and What Cannot Be Attributed **9**
5.1 Telemetry boundaries (`L0-L4`) and attribution labels (`A1-N`) . . . 9
5.2 Seven-step counter discipline, rootless collection (`OQ-5`), and platform maps . . . 10
5.3 Eight hardware measurement pitfalls and seven non-attributable quantities . . . 11

6. Measured Conformance, Determinism, Observer-Effect, and Multi-Host Attribution Results **12**
6.1 `C1-C12` conformance, `C9` deterministic replay, and `C10` process-tree cleanup . . . 12

6.2 Live C12 eight-request determinism probes across four model substrates . . . 12
6.3 Falsifier 5: Observer-effect benchmark across telemetry cadences (10 s windows) . . . 13
6.4 Falsifier 3 & Step 7: B temporal boundary vs. C proportional (Kepler) attribution . . . 13
6.5 Multi-session Falsifier 6 audit (188 windows) and Falsifier 8 harness-floor check . . . 14
6.6 Multi-platform hardware attribution across five silicon and OS archetypes . . . 14

7. Open Hardware Verification and Limitations **16**

References **16**

Revision 1561a2a, 22 November 2026. Data and digest summary: /data/#satb-harness. Interactive living figures: /living/satb-harness/. LaTeX archival twin: /pdfs/latex/satb-harness.pdf (source).

0. Overview

When two agent evaluations report different task-completion rates or different energy figures, the gap frequently comes from how the model was wrapped and how the host counters were attributed rather than from the model itself. Changing only the surrounding harness shifted Gemini 2.5 Pro task resolution by 17 percentage points on Terminal-Bench 2, while subtracting a drifted idle baseline or splitting a shared silicon rail by processor time can distort energy attribution by more than 30%.

This study specifies and empirically verifies the **Satiation Benchmark (SATB)** evaluation contract: one fixed sequential Rust reference harness (**Lane B**), one proxy-isolated adapter bridge (**Shapes D, M, H, and I**) for zero-model controls, raw model endpoints, external command-line harnesses, and public benchmark solvers, a five-block system-under-test (SUT) cryptographic fingerprint, twelve mandatory conformance checks, and a five-level rootless telemetry engine. Rather than leaving the design unmeasured, we execute live conformance and falsifier benchmarks across **five hardware archetypes** spanning macOS, Linux, and Windows 11: an Apple M5 Max workstation (128 gigabytes of unified memory, macOS 27.2.1, on-chip IOReport and kernel proc_pid_rusage reported_j), a Tegra aarch64 developer kit (Jetson Thor, Linux 6.8.12-1021-tegra, physical Texas Instruments whole-board and three-channel rail-separated current-shunt measured_j), a Framework Desktop Strix Halo workstation (Ryzen Max+ 395, Linux 7.0.0-43-generic x86-64, 0440 root:energy group-isolated package-energy reported_j plus amdgpu package-power tracking), a Raptor Lake workstation (Intel Core i9-13900H, Linux 7.0.0-43-generic x86-64, platform and package energy reported_j), and a bare-metal Windows 11 workstation (Intel Core i7-1165G7 plus GeForce 2060, Windows 11 build 26300 x86-64, rootless cumulative discrete-graphics reported_j, Win32 Job Object confinement, and unprivileged processor-package energy strictly marked not_available), alongside closed cloud server energy (not_available).

In plain words

1. An agent's prompt loop, context-trimming rule, and tool sandbox can swing benchmark scores as much as swapping the underlying model, so Lane B holds that wrapper completely fixed in deterministic Rust while Lane O forces external wrappers through one audited proxy plug.
2. Energy channels are recorded raw at their true physical boundary — whole wall plug, board input, chip package, operating-system process estimate, or logical token ledger — and are never blended into a single unlabelled number, summed across evidence classes (measured_j vs. reported_j vs. est_j), or back-filled with fleet sustainability averages.
3. Live tests across five silicon and operating-system platforms (macOS, Linux, and Windows 11) verify exact Shape D million-record memory-mapped (15027705736) and C9 replay (149,747 B, 0 newline or byte diffs) determinism, 0.87–9.54 ms kernel process-tree cleanup (2.46 ms via Win32 Job Objects on Windows 11), cloud temperature = 0 non-determinism across three vendors, hardware sensor cadence ceilings (24.3–45.4 µs Linux counter reads at 94–97 Hz vs. 1.17 ms Windows in-process graphics-counter reads at 64.6 Hz vs. 2.95 ms serial-bus shunt reads at 66.6 Hz on Jetson Thor vs. ~8 Hz macmon IOReport), 32.59%–89.88% error when splitting sequential compute-plus-storage phases by processor time (Falsifier 3), and physical graphics-versus-processor shunt rail separation alongside Falsifier 6 shared-host drift gating (dedicated_host = false).

1. Terms, acronyms, and attribution labels

- SATB, SUT, PFRK, OQ, SPEC, BUNDLE, SUBMIT, REFUSE, STOP: Satiation Benchmark harness and attribution specification, system under test, four-way attempt outcomes (pass, fail, refuse, killed), resolved specification open question, normative specification document, task bundle file, terminal answer submission call, terminal refusal call, and sequential loop halt token
- ACI, SWE, CLI, MCP, SDK, ACP, ARC, AGI, GUI: Agent-Computer Interface, software-engineering benchmark prefix, command-line interface, Model Context Protocol, software development kit, Agent Client Protocol, Abstraction and Reasoning Corpus, artificial general intelligence benchmark suite, and graphical user interface
- IO, JSON, JSONL, RPC, JCS, UTF, HTTP, SSE, URL: input/output protocol, JavaScript Object Notation, line-delimited JSON, remote procedure call, JSON Canonicalization Scheme (RFC 8785), Unicode Transformation Format, Hypertext Transfer Protocol, Server-Sent Events, and Uniform Resource Locator
- POSIX, MMAP, DNULLMM1, SHA, SIGKILL, ESRCH, EACCES, EOF, NT, NTFS, CRLF, MSVC, UCRT, FFI, DLL: Portable Operating System Interface, memory-mapped file interface, binary memory-mapped Deterministic Null index format, Secure Hash Algorithm (SHA-256), non-catchable POSIX process termination signal, kernel no-such-process status, POSIX permission-denied error status, end-of-file heredoc delimiter, Windows New Technology kernel (`Windows NT`), New Technology File System, carriage-return line-feed (`\r\n`) newline sequence, Microsoft Visual C++ compiler (`cl.exe`), Windows Universal C Runtime (`ucrtbase.dll`), foreign function interface, and dynamic-link library
- AI, ML, LLM, MLX, GGUF, KV, TP, ENERGY: artificial intelligence, machine learning, large language model, Apple array framework for local inference, GPT-Generated Unified Format quantized weights, key-value cache, tensor parallelism, and the ML.ENERGY GPU power benchmark consortium
- CPU, GPU, APU, ANE, SoC, ALU, ISA, IPC, PID, UID, GID, OS, VM, EC, GB, TGL, RTX: central processing unit, graphics processing unit, accelerated processing unit, Apple Neural Engine, system-on-chip, arithmetic logic unit, instruction set architecture, instructions per cycle, process identifier, user identifier, group identifier, operating system, virtual machine, embedded controller, gigabytes, Intel Tiger Lake mobile architecture, and NVIDIA GeForce RTX discrete GPU family
- RAM, DRAM, SLC, NVMe, PSU, PDU, AC, DC, DVFS, MSR, MMIO, RDTSC, HVCI, WMI, EMI, SRUM: random-access memory, dynamic RAM, system-level cache, Non-Volatile Memory Express, power supply unit, power distribution unit, alternating current, direct current, dynamic voltage and frequency scaling, model-specific register, memory-mapped input/output, x86 Read Time-Stamp Counter instruction, Windows Hypervisor-Protected Code Integrity, Windows Management Instrumentation, Windows Energy Metering Interface (`\\.\EnergyMeter`), and Windows System Resource Usage Monitor
- RAPL, NVML, PPT, TI, I2C, ADC, INA, INA238, INA3221, AGX, VDD, SOC, MSS, VIN, SYS, PLATYPUS: Intel/AMD Running Average Power Limit, NVIDIA Management Library, AMD Package Power Tracking, Texas Instruments, Inter-Integrated Circuit bus, analog-to-digital converter, Texas Instruments current/power monitor family (`INA238`, `INA3221`), NVIDIA Jetson AGX module tier, drain supply voltage rail, system-on-chip rail, memory subsystem rail, input system voltage rail, and Power Leakage Attacks: Targeting Your Processor Using Software
- API, USD, CNCF, AISI, UK, AMD, NVIDIA: application programming interface, United States dollars, Cloud Native Computing Foundation, AI Safety Institute, United Kingdom, Advanced Micro Devices, and NVIDIA Corporation
- RCE, MoL, WCA, LUT6: Reality Code Engine [same house], Mixture of Limits [same house], Wise Computer Automation [same house], and 6-input Boolean look-up table

Across telemetry levels L0 to L4, every channel recorded in a `SATB` receipt carries both its physical boundary and one of four strict evidence classes:

Evidence class	Scope in `SATB` receipts and tables
`Exact (L4 Logical & Cryptographic)`	`SHA-256` and `JCS` (`RFC 8785`) canonical digests of task bundles, five-block `SUT` fingerprints, and replay transcripts; proxy-logged prompt, cache, reasoning, and completion token counts; wall-clock durations; and dated `API` list-price `USD`.

Evidence class	Scope in SATB receipts and tables
Measured Hardware Rail (measured_j: L0, L1, L2)	Populated only from a calibrated external AC wall analyzer (L0) or physical board current-shunt monitor (L1/L2). Verified live on NVIDIA Jetson AGX Thor via TI INA238 (hwmon6 VIN, L1) and INA3221 (hwmon5 ch1 VDD_GPU, ch2 VDD_CPU_SOC_MSS, ch3 VIN_SYS_5V0, L2/L1; board_metered = true). Because an active co-tenant held VDD_GPU during capture (dedicated_host = false), raw rail measured_j is preserved while A2 net placement attribution is withheld; measured_j remains None on Apple M5 Max, AMD Strix Halo, Intel Raptor Lake, and Windows 11 (i7-1165G7 + RTX 2060), and L0 wall power is unset throughout.
Reported Counter (reported_j: L1, L2 & L3)	Rootless on-chip telemetry counters (Apple M5 Max IOReport sys_power at L1 and chip_power/cpu_power/gpu_power at L2 via macmon; Intel Raptor Lake RAPL psys at L1 and package-0/core/uncore at L2; Framework AMD Strix Halo RAPL package-0/core at L2 plus amdgpu PPT watts; Windows 11 rootless NVML nvmlDeviceGetTotalEnergyConsumption cumulative mJ on NVIDIA GeForce RTX 2060 at L2) and OS-modelled per-process estimates (macOS proc_pid_rusage(RUSAGE_INFO_V6).ri_energy_nj at L3, label D). Never summed with measured_j or est_j.
Provider-Attested (P) vs. Not Available (N)	Closed-API remote server energy (and Windows 11 user-mode CPU package energy where ring-0 RAPL MSR drivers are barred) is written as P only when a provider returns a per-request energy figure with verbatim methodology, and otherwise literally as "not_available" (label N). Derived baseline subtractions failing the 5.0% drift bound (Falsifier 6) or the Step 7 sum check are automatically dropped while raw channels remain intact.

1.1 Lineage, design invariants, and same-house disclosure

SATB builds directly on public agent-evaluation and power-attribution work rather than inventing bespoke protocols where standards exist: minimal stateless shell execution follows mini-swe-agent; solver and external-CLI bridging draw on UK AISI Inspect (react() and Inspect Agent Bridge) and Terminal-Bench (Harbor / Terminus-2); tool schemas use MCP canonicalized via RFC 8785 (JCS); GPU cumulative counter discipline follows Chung et al. (ML.ENERGY / Zeus) and Yang et al. on nvidia-smi part-time sampling pitfalls; proportional CPU splitting (label C) is evaluated against the CNCF Kepler model; and Linux RAPL permission isolation follows Lipp et al. (PLATYPUS). Companion Open Interface Engineering studies — Metered Commit Gates, Task Satiation vs. Aggregate Rebound, and Deterministic Null — are [same house] works cited strictly for shared architectural definitions:

Contract axis	Invariant preserved across all lanes
Attempt protocol	Fixed task ladder; at most one terminal SUBMIT or REFUSE per attempt; one immutable SHA-256 receipt per attempt; canonical outcomes pass, fail, refuse, killed; zero verifier feedback during test attempts.
System identity	Five-block SUT fingerprint (model + serving + sampling + harness + envelope), hashed at session.open and re-hashed at session.close.
Attributable energy	Whole-SUT raw window energy (A2) on a quiesced dedicated host; inference-engine energy on an exclusive physical rail (A1); sequential model-vs-tool phase energy (B) under Lane B.
Non-attributable	Unattested closed-API server joules (N); exact per-process joule shares of a shared SoC or unified-memory rail (C and D are models, not physical rail meters); model weights divorced from engine kernels and silicon.

---

2. Survey of Agent Harnesses and Why Context Policy Is Part of the System

Modern coding, terminal, and computer-use harnesses differ sharply in how they expose tools, gate file edits, and compact growing conversation histories:

Harness	Loop and tool surface	Context and history policy
mini-swe-agent	Bash only. Every action spawns a fresh stateless subprocess.run.	Strictly linear append-only message history; no automatic compaction.
SWE-agent (ACI)	100-line windowed file viewer, linter-gated edit command, succinct search.	History processor folds older environment observations to bounded summaries.

Harness	Loop and tool surface	Context and history policy
`Inspect react()`	Sequential tool loop until `submit()`: `bash()`, `text_editor()`, `MCP` tools.	Editable message state with pluggable history truncation or compaction solvers.
`Inspect Agent Bridge`	Wraps `Claude Code`, `Codex CLI`, `Gemini CLI`, and custom `SDK` agents behind a local proxy.	Delegates context management entirely to the wrapped external harness.
`Terminus-2 (Harbor)`	Single tool: persistent `tmux` terminal session receiving raw keystrokes.	Three-stage context summarization (`3-stage` progressive transcript compression).
`Codex CLI`	Sandboxed shell (`Landlock`/`seccomp` or `seatbelt`), structured `apply_patch`, `MCP`.	Native rolling context compaction triggered near model window limits.
`Claude Code`	`Bash`, structured file tools (`Read`, `Write`, `Edit`, `Glob`, `Grep`), `MCP`.	Automatic LLM-driven conversation compaction (`auto-compact`) plus prompt caching.
`OpenHands`	Event-stream runtime executing any `ACP` or `CodeAct` coding agent in containers.	Configurable context condenser (`LLM` summarization or window masking).
`ARC-AGI-3 runner`	Standard harness vs. Provider Adapter harnesses over interactive grid states.	Bounded visual/grid state trajectory window declared per adapter track.
`OSWorld run.py`	Desktop `GUI` loop emitting `pyautogui` mouse/keyboard actions over VM screenshots.	Sliding trajectory window retaining only the most recent K screenshots and actions.

On `Terminal-Bench 2`, `Gemini 2.5 Pro` resolved **`17% more tasks`** when paired with `Terminus-2` than when paired with `OpenHands` — holding the underlying model checkpoint constant. Context management, output truncation, edit linting, and retry rules are therefore first-order components of the evaluated system rather than neutral plumbing. `SATB` resolves this confound by splitting evaluation into two explicit lanes and closing Open Questions `OQ-1` and `OQ-2`:

1. **Holding the wrapper constant (`Lane B`) vs. evaluating external wrappers (`Lane O`)**: `Lane B` (`satb-agent`) fixes a single auditable Rust loop across all models, whereas `Lane O` wraps complete third-party harnesses (`Claude Code`, `Codex CLI`, `OpenHands`, `Terminus-2`, `Inspect`) as black-box `SUT`s whose internal prompts and compaction calls are hashed into the `harness` fingerprint block and routed 100% through `satb-proxy`.
2. **Context policy resolution (`OQ-1: ContextRule_v0`)**: Hard-killing an attempt with zero elision conflates multi-step reasoning with verbose tool output, whereas `LLM`-generated summarization (`Claude Code` auto-compact) injects hidden stochastic model calls that break deterministic harness replay. `Lane B` adopts `ContextRule_v0`: whenever accumulated conversation size exceeds **`80%`** of the declared context window, the harness deterministically replaces the oldest tool outputs with `"[ELIDED_BY_CONTEXT_RULE_V0]"` while keeping `system(TEMPLATE_v0)`, the initial `user(render(bundle))`, and the **most recent `8` tool outputs** verbatim, making **zero model calls**. Hard `killed(context_ceiling)` fires only if those retained messages alone exceed `100%` of the window.
3. **Tool surface resolution (`OQ-2: TOOLS_v0`)**: Pure `exec`-only (`mini`-style) forces models to shell-escape multi-line source patches and algebraic proofs (`cat << 'EOF'`), introducing shell-quoting failures orthogonal to task reasoning and complicating portable rootless sandboxing across `macOS`, `Linux`, and `Windows`. `Lane B` fixes `TOOLS_v0` — four workspace tools (`exec`, `read_file`, `write_file`, `list_dir`) plus two terminal tools (`submit`, `refuse`), canonicalized via `JCS` into `tools_hash` (`d5b102...`).

3. Reference Harness (Lane B), ContextRule_v0, and SATB-IO Protocol

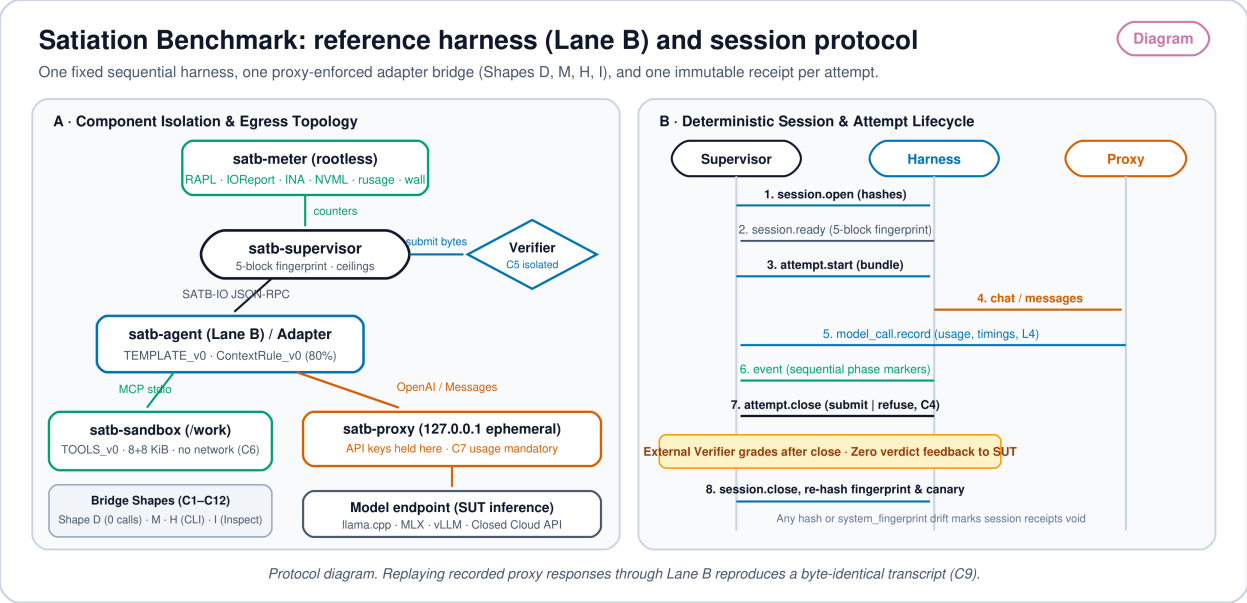

Figure 1. Left: Lane B component isolation separating rootless satb-meter, satb-supervisor, satb-agent, network-blocked satb-sandbox, credential-holding satb-proxy, and the unreachable external Verifier. Right: deterministic SATB-IO session and attempt sequence.

3.1 Sequential execution loop and fixed-versus-configurable split

Inside satb-agent, every attempt executes a strictly sequential single-agent loop with no overlapping background tool threads, no best-of-k voting, and no verifier oracle feedback:

```
msgs <- [system(TEMPLATE_v0), user(render(bundle))]
loop:
  ContextRule_v0(msgs)        # >= 80% window -> elide oldest tool outputs; keep last 8; 0
    model calls
  r <- proxy.chat(msgs, TOOLS_v0)   # missing usage -> immediate killed(usage_missing)
  if no tool calls: append CONTINUE_v0; continue
  for call in r.tool_calls (sequential):
    submit | refuse -> attempt.close; ignore trailing calls (C4); STOP
    exec | read_file | write_file | list_dir -> fresh sandboxed process in /work,
      Truncate_v0 (8 KiB head + 8 KiB tail), no network egress (C6)
  envelope ceiling reached -> killed(wall_ceiling | token_ceiling | turn_ceiling |
    context_ceiling)
```

Every Lane B run separates fixed benchmark constants (hashed into harness_id) from declared SUT parameters (hashed into sut_fingerprint_sha256):

Fixed across all Lane B runs (harness_id)	Configurable per SUT (sut_fingerprint_sha256)
TEMPLATE_v0 prompt, bundle renderer, and CONTINUE_v0 nudge	model_id, provider snapshot date, or local GGUF/safetensors SHA-256
TOOLS_v0 schemas (tools/list JCS digest) and sequential dispatch	Sampling parameters: temperature, top_p, top_k, seed, max_tokens, reasoning_effort
ContextRule_v0 (80% threshold, keep last 8) and Truncate_v0 (8 + 8 KiB)	Serving engine (llama.cpp, MLX, vLLM, cloud API) and batch/cache flags
Single attempt (attempts = 1), zero verifier feedback, zero internal voting	Envelope ceilings: wall_ms, max_total_tokens, max_turns
satb-sandbox: /work confinement, stripped environment, zero network egress	Sandbox CPU/RAM quotas, host silicon, OS kernel, and isolation tier

3.2 Canonical four-way outcome mapping (OQ-3)

To prevent benchmarks from silently dropping crashed runs or conflating voluntary bad answers with infrastructure timeouts, SATB maps legacy five-state completion codes losslessly onto **{pass, fail, refuse, killed}** (PFRK), preserving the exact sub_reason in every receipt:

1. **completed → pass (sub_reason: "verified", score 1)**: SUT called submit(bytes) prior to ceiling expiration and the external Verifier accepted bytes after attempt.close.
2. **incorrect → fail (sub_reason: "incorrect", score 0)**: SUT called submit(bytes) cleanly; external Verifier rejected the candidate.
3. **malformed → fail (sub_reason: "malformed_submission", score 0)**: SUT voluntarily called submit(bytes) with unparseable or schema-invalid content. A bad submission authored by the SUT is an accuracy failure, never an infrastructure kill.
4. **refused → refuse (sub_reason: "policy" | "capability" | "other", score 0)**: SUT explicitly invoked refuse(reason).
5. **no_submission → killed (sub_reason: "wall_ceiling" | "token_ceiling" | "turn_ceiling" | "context_ceiling" | "usage_missing" | "exec_timeout_killed" | "process_crash", score 0)**: satb-supervisor terminated the attempt before any terminal tool call arrived. All four outcomes remain in the denominator n.

---

4. Adapter Bridge (Shapes D, M, H, I), Five-Block SUT Fingerprint, and C1–C12 Conformance

4.1 Four adapter shapes behind one supervisor and proxy

Any system outside Lane B connects to satb-supervisor through one of four adapter shapes:

Adapter shape	Target systems	Execution and isolation contract	Standards reused
Shape D (dnull control)	Deterministic Null (N0 strict model-free and N1 dev-crystallized) over frozen DNULLMM1 mmap stores	Executes inside /work under identical TOOLS_v0 and Verifier rules with 0 **proxy model calls** at decision time.	POSIX mmap / Win32 MapViewOfFile, SPEC v0.4 64-bit integer loop
Shape M (Model endpoint)	Local open-weight servers and closed cloud APIs	Runs Lane B satb-agent over Chat Completions or Messages with tool calls and mandatory usage, routed 100% via satb-proxy.	OpenAI-compatible API (llama.cpp, mlx_lm, vLLM); Anthropic, Gemini, xAI translators
Shape H (Harness in a box)	Claude Code, Codex CLI, Gemini CLI, OpenHands, Terminus-2	Task written to /work/task/BUNDLE.md. Base URL pointed at satb-proxy (127.0.0.1:<port>); real API keys withheld from /work; ends via MCP submit or refuse; zero other egress.	Inspect sandbox bridge env vars; --strict-mcp-config MCP; optional ACP
Shape I (Inspect agent)	Existing public benchmark Inspect solvers	Wraps SATB task bundle with submit and refuse tools at attempts = 1, routing model calls through satb-proxy.	UK AISI Inspect Task and Solver interface

4.2 Five-block SUT fingerprint and session validity

Before attempt.start can fire, the adapter returns a five-block JSON structure whose canonical JCS SHA-256 digest defines the SUT. Unknown or unobservable fields must be written literally as "not_available", never omitted or guessed:

Fingerprint block	Required fields (unavailable values written literally as "not_available")
model	provider, model_id, snapshot_date, weights_sha256, quantisation, tokenizer_hash, chat_template_hash, context_window, system_fingerprint
serving	engine, version, commit, binary_hash, flags_hash (cache_prompt, slots, batch_invariance, TP), drivers, devices
sampling	temperature, top_p, top_k, seed, max_tokens, stop, reasoning_effort, tool_choice

Fingerprint block	Required fields (unavailable values written literally as "not_available")
harness	name, version, commit, binary_hash, prompt_hash, tools_hash, context_policy, internal_voting, sub_agents, config_hash
envelope	hardware, os_kernel, isolation_tier, cgroup_limits, cpuset, network_policy, ceilings, telemetry_config_hash

Session voiding and start/end canaries. Any changed block digest at session.close, any MCP tools/list_changed notification, an inference-engine restart, an NVML counter reset, a changed echoed model or system_fingerprint mid-session, or an OS/driver update marks **every receipt in the session void**. Void receipts remain in the public archive for auditability but are excluded from benchmark placement. Because a cloud provider can swap weights or quantization behind a fixed model_id without updating system_fingerprint, every session also brackets evaluation with a deterministic canary rung at session.open and session.close.

4.3 Mandatory C1-C12 adapter conformance suite

No adapter or harness enters evaluation until it passes all twelve conformance checks (C1 to C12):

Check	Requirement for passing
C1 **Bundle round-trip**	Adapter echoes the exact JCS SHA-256 digest of the input task bundle.
C2 **Submit path**	External Verifier receives the exact byte sequence logged at satb-proxy / MCP terminal capture.
C3 **Refuse path**	An instructed refusal arrives cleanly as refuse (sub_reason = "other", completion_score = 0).
C4 **One terminal call**	Any tool or terminal calls emitted after the first submit or refuse are ignored and counted.
C5 **No oracle**	Verifier and satb-supervisor directories, sockets, and parent paths (..) are unreachable from /work.
C6 **Egress**	Only satb-proxy (127.0.0.1:<port>) is reachable for model traffic; /work subprocesses have zero external network access and zero upstream API keys.
C7 **Usage present**	Every model call carries usage (missing -> killed(usage_missing)); proxy token totals match upstream engine counts exactly.
C8 **Fingerprint stable**	Five-block SUT fingerprint SHA-256 is identical before and after a 3-attempt dry run.
C9 **Harness replay**	Replaying recorded satb-proxy responses reproduces a byte-identical transcript (0 byte or CRLF diffs; mandatory for Lane B).
C10 **Kill path**	Runaway /work process tree (POSIX process group or Win32 Job Object, including detached child processes) is reaped within < 2,000 ms and recorded as killed.
C11 **Telemetry handshake**	Rootless L3 process counters (cgroup v2, proc_pid_rusage, or Win32 GetProcessTimes/QueryProcessCycleTime) are readable, phase markers arrive, and monotonic-to-wall clock offset is recorded.
C12 **Determinism probe**	8 identical requests are executed and the count of distinct outputs is recorded as an empirical SUT property.

5. Five-Level Telemetry Attribution (L0–L4 × A1–N), Platform Channel Maps, and What Cannot Be Attributed

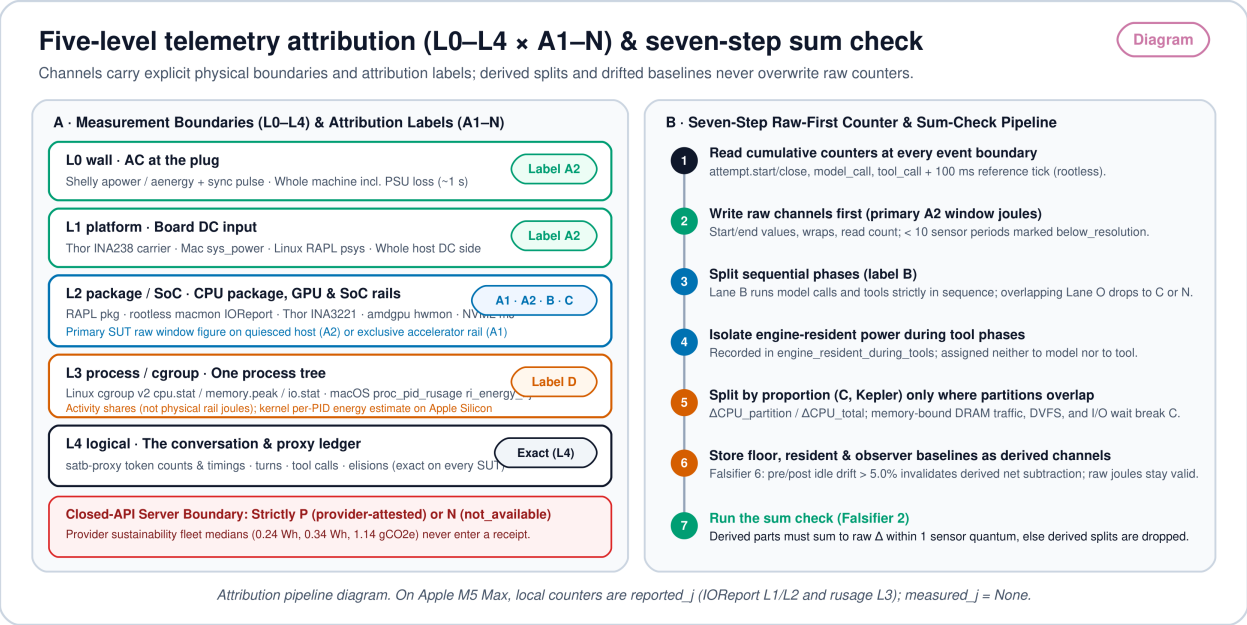

Figure 2. Left: physical and logical telemetry levels L0–L4 mapped to attribution labels A1–N. Right: seven-step raw-first counter discipline enforcing pre/post baseline drift bounds (Falsifier 6) and the Step 7 sum check.

5.1 Telemetry boundaries (L0–L4) and attribution labels (A1–N)

Energy and activity sensors sit at five distinct boundaries that measure physically different quantities and must never be summed or conflated:

Level	Physical / logical boundary	Representative sensors	What the channel can honestly attribute
L0 wall	AC at the wall receptacle	Smart plug or PDU (Shelly apower, aenergy; Yokogawa analyzer)	Whole machine including PSU conversion heat, fans, DRAM, and NVMe; ~1 s cadence.
L1 platform	Board DC input rail	Jetson Thor INA238 carrier; Mac sys_power; RAPL psys	Whole computer on the DC side downstream of the power brick or PSU.
L2 package / SoC	CPU package, GPU, and SoC rails	RAPL pkg; rootless macmon IOReport; Thor INA3221; amdgpu hwmon; NVML cumulative energy (mJ)	Whole SUT on SoC (A2), or inference engine alone when a rail is physically exclusive (A1).
L3 process / cgroup	Single OS process tree	Linux cgroup v2 (cpu.stat, memory.peak, io.stat), perf; macOS proc_pid_rusage ri_energy_nj; Win32 GetProcessTimes + QueryProcessCycleTime	Resource activity shares (C), or the OS kernel's own internal per-process energy model (D).
L4 logical	The model/tool conversation	satb-proxy token counts and call latencies; harness turns, tool calls, and ContextRule_v0 elisions	Exact counts on every SUT without exception, including closed cloud APIs.

Every energy value in a receipt carries one of seven attribution labels:

- **A1 (Exclusive physical rail)**: A hardware rail sensor touched exclusively by one partition during the measurement window (e.g., Jetson Thor INA3221 ch1 VDD_GPU when only the local inference engine has GPU device access).
- **A2 (Dedicated whole-system boundary)**: A whole-machine (L0/L1) or whole-package (L2) counter on a quiesced dedicated machine (SUT plus idle OS floor). **Per OQ-4, whole-SUT raw window energy at A2 is the primary physical energy metric for placement**, preventing thin models from offloading uncounted brute-force search into /work subprocesses and preventing active - idle subtraction from flattering slow runs.

- `B` **(Temporal phase split)**: Counter differences across non-overlapping time windows delimited by sequential phase markers (`model_call` vs. `tool_call` in `Lane B`).
- `C` **(Proportional share — the Kepler model)**: A shared hardware counter partitioned by $\Delta\text{CPU}_{\text{partition}}/\Delta\text{CPU}_{\text{total}}$ or `GPU` utilization when parallel processes overlap (`Lane O Shape H`).
- `D` **(OS-modelled estimate)**: The kernel's internal per-process energy model (`macOS ri_energy_nj`). Isolates a process tree from `CPU` co-tenants, but remains a firmware/kernel model rather than a physical shunt reading.
- `P` **(Provider-attested)**: A per-request server energy number returned by a cloud provider alongside its verbatim measurement methodology.
- `N` **(not_available)**: Unobserved and **never estimated**.

5.2 Seven-step counter discipline, rootless collection (OQ-5), and platform maps

`satb-meter` processes every attempt through seven ordered steps:

1. **Read cumulative counters at every event boundary** (`attempt.start`, each `model_call` and `tool_call`, `attempt.close`, plus a `100 ms` reference tick) so window energy is an exact counter difference $\Delta E = E(t_{\text{end}}) - E(t_{\text{start}})$.
2. **Write raw channels first**: record boundary, sensor, start/end readings, counter wraps, and sample count; mark windows shorter than `10` sensor periods `"below_resolution"` and absent hardware `"not_available"`.
3. **Split sequential phases (`B`)**: in `Lane B`, model inference and `/work` tool calls never overlap; parallel `Lane O` harnesses drop phase splits to `C` or `N`.
4. **Isolate `engine_resident_during_tools`**: while a `/work` tool runs, a local model server (`llama-server`, `vLLM`) holds weights resident on the `GPU`/`DRAM`. Its idle draw during the tool window is placed in `engine_resident_during_tools` and charged neither to model active inference nor to tool execution.
5. **Split by proportion (`C`) only where partitions overlap**: apply ΔCPU ratios strictly as a characterisation fallback, recognizing that `C`-states, turbo `DVFS`, and `DRAM`/storage stalls violate proportionality.
6. **Store baselines as derived channels**: bracket sessions with pre/post resident baselines; if pre/post idle power drifts beyond `5.0%` (`Falsifier 6`), drop derived net-above-idle values while keeping raw channels intact.
7. **Run the sum check (`Falsifier 2`)**: verify that derived phase components sum back to the raw attempt counter delta within one sensor quantum (`1 mJ`); if they do not, discard the derived split and retain only the raw channel.

Rootless collection rule (`OQ-5`). Benchmarks must never run `SUT` orchestration as `root` or load custom ring-0 drivers. On `macOS`, `satb-meter` uses rootless `macmon` (`IOReport` subscription) and `libSystem proc_pid_rusage(RUSAGE_INFO_V6)`, never `sudo` or `powermetrics`. On `Linux` (where `/sys/class/powercap/intel-rapl` became root-only in kernel `5.10` following `PLATYPUS`), a one-time administrator `udev` rule grants read access to a dedicated `satb-meter` group containing **only `satb-supervisor`'s `UID`**, ensuring `/work` sandbox processes can neither side-channel `RAPL` nor read their own energy meter. On `Windows 11` (`NT 10.0.26300`), unprivileged Basic User tokens (`0x20000`) read discrete-`GPU` cumulative energy (`mJ`) directly via `nvml.dll` (`nvmlDeviceGetTotalEnergyConsumption`) and `L3` process cycles via Win32 `QueryProcessCycleTime`, whereas Intel `RAPL` `MSR` `0x611` requires a ring-0 kernel driver barred by `OQ-5` and `HVCI` (placing Windows `CPU` package energy strictly in `"not_available"`, label `N`):

Target platform	`L0` / `L1` whole-system channels	`L2` package / `SoC` channels	`L3` process & `L4` logical channels	Key verification status
Apple Silicon Mac (`M5 Max` verified)	`L0`: `Shelly PDU` (`not_available` if unplugged); `L1`: `macmon sys_power` (`reported_j`)	`macmon IOReport`: `chip_power`, `cpu_power`, `gpu_power`, `ane_power`, `ram_power` (`reported_j`)	`L3`: `proc_pid_rusage ri_energy_nj` (`D`), `ri_instructions`, `ri_diskio_*`; `L4`: `satb-proxy`	Rootless `L1`–`L4` verified live on `Apple M5 Max`; `L0` plug open.
Framework AMD Strix Halo (`Ryzen AI Max+ 395`, `Linux x86-64` verified)	`L0`: `Shelly PDU` (`not_available` if unplugged); `L1`: `cros_ec` (0 power attrs)	`RAPL package-0` & `core` (`0440 root:energy udev`) `reported_j` + `amdgpu PPT power1_input` (`0444`; `energy1_input` absent)	`L3`: `cgroup v2` / `getrusage` (`C` activity shares); `L4`: `satb-proxy`	Verified live (`0440 root:energy PLATYPUS` isolation; `amdgpu PPT 0444`).

Target platform	L0 / L1 whole-system channels	L2 package / SoC channels	L3 process & L4 logical channels	Key verification status
Intel Raptor Lake (Core i9-13900H, Linux x86-64 verified)	L0: not_available; L1: RAPL psys (intel-rapl:1/energy_uj, 0444 reported_j)	RAPL package-0, core, & uncore (MSR + MMIO energy_uj, 0444 reported_j)	L3: cgroup v2 / getrusage (C activity shares); L4: satb-proxy	Verified live (0 µJ Step 7 residual across psys L1 and package-0 L2).
NVIDIA Jetson AGX Thor (Linux aarch64 Tegra verified)	L1: TI INA238 (hwmon6 VIN, 0444 measured_j) + INA3221 ch3 VIN_SYS_5V0 (0444 measured_j)	INA3221 ch1 VDD_GPU (0444 measured_j, A1 when exclusive); ch2 VDD_CPU_SOC_MSS (0444 measured_j); NVML (unverified)	L3: cgroup v2 / getrusage (C activity shares); L4: satb-proxy	Verified live (INA238 + INA3221 ch1-ch3 0444 rootless; GPU NVML open under co-tenant lock).
Windows 11 x86-64 + Discrete RTX (i7-1165G7 + RTX 2060, NT 26300 verified)	L0: not_available; L1: WMI EMI PowerMeter (0 instances -> not_available)	RTX 2060: rootless nvml.dll cumulative mJ (reported_j); CPU RAPL MSR 0x611: **N (not_available without ring-0 driver)**	L3: Win32 GetProcessTimes + QueryProcessCycleTime (C shares; SRUM coarse OS background excluded); L4: satb-proxy	Verified live under Basic User 0x20000 token (0 mJ Step 7 residual; 2.46 ms Job Object kill).
Closed Cloud API (Anthropic, xAI, Cerebras, OpenAI, Google)	Local L0–L1 measure client host only (excluded from server inference)	Remote server L2: strictly N (not_available) or per-request P (OQ-6)	Local /work L3 (D/C) for sandbox tools; L4: **exact** satb-proxy tokens, calls, USD	Verified live across Anthropic, Cerebras, and xAI via satb-proxy.

5.3 Eight hardware measurement pitfalls and seven non-attributable quantities

Standard power-measurement shortcuts fail systematically on modern SoCs and GPUs:

Hardware telemetry pitfall	SATB mitigation
Idle power drifts with die temperature, fans, DVFS, and turbo	Wait for GPU cooldown (<= 1.5 W), bracket every window with pre/post baselines, record peak GPU temperature, and invalidate derived net joules above 5.0% drift (Falsifier 6).
Ambiguity between cold-machine idle and weights-resident idle	Measure and label both explicitly in Step 6 rather than silently subtracting one.
Active - idle subtraction flatters slow runs	Whole-SUT raw window energy (A2) remains the primary placement figure (OQ-4).
Sub-period windows read as zero, negative, or quantization noise	Mark any window shorter than 10 sensor periods "below_resolution" (e.g., < 1.2 s on ~8 Hz macmon or < 10 s on a 1 Hz smart plug).
High-rate telemetry samplers burn significant CPU power	Read cumulative counters at event boundaries (0.011 mW at 0 Hz) plus a 100 ms tick; record observer_only baseline.
NVML instantaneous power gauge on A100/H100 samples only ~25% of runtime	Read cumulative nvmlDeviceGetTotalEnergyConsumption (mJ) via in-process FFI (1.17 ms on RTX 2060 vs. 43.88 ms nvidia-smi subprocess), never instantaneous nvmlDeviceGetPowerUsage polling (Chung et al.; Yang et al.).
External L0 wall plug runs on an unsynchronized clock	Emit a deterministic 50%-duty calibration load pulse at session.open and session.close and cross-correlate timestamps.
Shared-host background daemons or co-tenant jobs	Log process list, downgrade shared L2 derived subtractions via Falsifier 6, and record co-tenant-isolated L3 ri_energy_nj (D) alongside raw L2.

Consequently, **seven quantities cannot be honestly attributed** and are barred from SATB claims: (1) unattested closed-API server energy (N or P only); (2) exact per-process physical joules on a shared SoC or unified-memory rail (C and D are models); (3) model weights isolated from serving kernels, quantization, batching, and silicon; (4) ownership of the OS idle floor and DRAM-resident weight power (reported as standalone channels); (5) individual events shorter than the sensor period (such as a 30 ms tool call under a 1 s plug); (6) shared cloud VMs or unprivileged Windows CPU domains lacking user-mode hardware counters; and (7) silent mid-session provider weight swaps (flagged by start/end canaries, never ruled out a priori). In particular (OQ-6), **provider sustainability disclosures are fleet medians rather than per-request measurements and never enter a SATB receipt** — including Google's 0.24 Wh median Gemini Apps prompt, OpenAI's 0.34 Wh figure (published without methodology), and Mistral's 1.14 gCO2e per 400-token response.

6. Measured Conformance, Determinism, Observer-Effect, and Multi-Host Attribution Results

To replace unmeasured design assumptions with auditable receipts, we executed the complete SATB conformance suite and falsifier harness (bench_satb_falsifiers.py → satb_empirical_receipts.json, total runtime 175.69 s) on an Apple M5 Max workstation (arm64, 128 GB unified memory, macOS 27.2.1, macmon 0.12.0), alongside a 188-window cross-study audit and live cross-OS multi-silicon runs across Linux and Windows 11.

6.1 C1–C12 conformance, C9 deterministic replay, and C10 process-tree cleanup

All 12 / 12 **Rust reference conformance checks** (dnull_cli satb-conformance) and all extended operating-system checks passed:

- **Bundle, terminal, confinement, and fingerprint checks (C1–C8, C11)**: C1 echoed canonical bundle digest 455e941f51f8ffef; C2 delivered byte-exact submit payloads across satb-proxy to Verifier; C3 mapped empty/instructed refusals to refuse:other; C4 ignored trailing post-submit tool calls (ignored_trailing_calls = 1); C5 blocked ../, /etc/passwd, and verifier_oracle/* escapes; C6 stripped proxy/credential environment variables inside /work (NP=*|HTTP=) while returning HTTP 404 on unlisted proxy paths; C7 aborted missing-usage responses immediately as killed(usage_missing); C8 preserved byte-identical five-block fingerprint 8b741c27bf364e5d across dry runs while flagging simulated system_fingerprint swaps with session_void = true; and C11 verified rootless RUSAGE_INFO_V6 counter visibility (sudo_invoked = false).
- **Extended C9 multi-turn ContextRule_v0 + Truncate_v0 replay**: Across a 14-turn trace generating 24 KiB raw file reads per turn (14 outputs truncated to 8,192 B + 8,192 B by Truncate_v0) inside a 48,000-token window (80% threshold = 38,400 tokens), ContextRule_v0 deterministically elided the **5 oldest tool outputs** with **0 extra model calls**, producing a 149,747-byte canonical transcript (SHA-256 d697ed14 76129939...) with **0 byte differences** across independent /work replays. Repeating the identical trace under a tight 16,000-token window (where 8 retained 16 KiB outputs exceed 100%) deterministically triggered killed(context_ceiling).
- **Extended C10 multi-process SIGKILL cleanup (< 2,000 ms specification ceiling)**: Across 5 trials spawning a 4-process tree (sh parent plus 3 detached background Python sleepers in an isolated process group), SIGKILL delivery and kernel reaping completed in **2.073 ms mean** (1.429 ms min, 2.571 ms max — **778× below** the 2,000 ms ceiling) with **0 surviving PIDs** and outcome killed(wall_ceiling).

6.2 Live C12 eight-request determinism probes across four model substrates

Routing 8 identical algebraic-synthesis requests per model (32 live cloud calls plus 8 local Shape D runs) through satb-proxy verified 100% exact C7 token-count parity (32/32 calls) and exposed three distinct determinism regimes:

SUT / Endpoint via satb-proxy	temperature = 0 preflight	Recorded sampling. temperature	Distinct raw outputs (/ 8)	Distinct algebraic forms (/ 8)	system_ fingerprint	Mean latency
Shape D: dnull N1 (Rust local)	Exact i64	0 (exact_i64_q1e6)	1 / 8	1 / 8	0da4df8d 9ccbce1e…	1,474.0 ms
Anthropic claude-opus-5-5	HTTP 400	not_available (deprecated)	8 / 8	4 / 8	not_available	3,389.1 ms
Anthropic claude-sonnet-5-5	HTTP 400	not_available (deprecated)	8 / 8	2 / 8	not_available	2,914.3 ms
Cerebras gpt-oss-120b	HTTP 200	0.0 (seed=42)	2 / 8	2 / 8	fp_fd4dff7e ed92aaf38281	484.2 ms
xAI grok-4.20-0309-non-reasoning	HTTP 200	0.0 (seed=42)	8 / 8	1 / 8	fp_9974b11849	942.7 ms

1. **Provider deprecation of temperature = 0**: Both claude-opus-5-5 and claude-sonnet-5-5 rejected "temperature": 0.0 with **HTTP 400 ("temperatureis deprecated for this model.")**, empirically validating SATB's rule that unavailable sampling controls be normalized to "not_available" rather than silently assumed to equal zero. Across 8 identical calls, claude-opus-5-5 (1,047 adaptive thinking tokens) emitted 8 distinct raw texts spanning 4 algebraically distinct bitvector identities ((x | y) - (x & y), (x | y) & ~(x & y), (x & ~y) | (~x & y), and (x + y) - ((x & y) + (x & y))).

2. **Residual non-determinism under `temperature = 0, seed = 42`**: Even when `temperature = 0.0` and `seed = 42` were accepted (`HTTP 200`) with a stable `system_fingerprint`, `Cerebras gpt-oss-120b` (`2,976` reasoning tokens) produced `2` distinct outputs across `8` identical calls, while `xAI grok-4.20-0309-non-reasoning` stabilized on `1` algebraic formula (`(x | y) & ~(x & y)`) across all `8` calls yet varied its prose proof sentence on all `8/8` responses. Separating **harness determinism given recorded responses (`C9`)** from **measured `SUT` output dispersion (`C12`)** is therefore essential.

6.3 Falsifier 5: Observer-effect benchmark across telemetry cadences (`10 s` windows)

Measuring combined sampler (`Python` + child `macmon IOReport` pipe) `CPU` time, `L3` `ri_energy_nj` (`observer_only reported_j`), and whole-`SoC` `L2` `reported_j` across interleaved `10.0 s` windows on `Apple M5 Max` quantified exact observer overhead:

Sampling cadence	Achieved rate	Sampler `CPU` time / `10 s`	Observer `L3` `reported_j`	Observer power (`mW`)	Share of `L2` window	Whole-`SoC` `L2` window (`reported_j`)
`0 Hz` **boundary-only** (`2` rusage syscalls)	`0.20 Hz`	`0.004 ms`	`0.000108 J`	`0.011 mW`	`0.0000%`	- (zero `IOReport` daemon)
`1 Hz (1,000 ms tick)`	`0.90 Hz`	`9.323 ms`	`0.194237 J`	`19.371 mW`	`0.0521%`	`372.769 J (37.17 W)`
`10 Hz (100 ms SATB` **reference tick)**	`8.12 Hz`	`36.434 ms`	`0.666596 J`	`66.439 mW`	`0.1770%`	`376.603 J (37.53 W)`
`100 Hz (10 ms stress` **falsifier)**	`7.44 Hz`	`35.636 ms`	`0.598721 J`	`58.971 mW`	`0.1618%`	`369.926 J (36.44 W)`

Two hardware facts emerge directly from this measurement:

1. **Negligible perturbation (`< 0.18%`)**: Boundary-only `proc_pid_rusage` reads consume `108 µJ` (`0.011 mW`), **`6,040×` less power** than `10 Hz` `IOReport` polling (`66.44 mW`), and even the `100 ms` reference tick represents just **`0.177%`** of whole-`SoC` window energy (`0.76%` total `L2` window difference between `-i 10` and `-i 1000`, clearing `Falsifier 5`).
2. **`IOReport` subscription cadence ceiling (`~8 Hz`)**: Requesting `-i 10` (`100 Hz`) from `macmon 0.12.0` achieved **`7.44 Hz`** (`75.5` samples in `10.15 s`) because each rootless `IOReport` subscription cycle blocks for `~120 ms` in macOS user space. Consequently, `Step 2`'s `10`-period `below_resolution` floor corresponds physically to **`~1.2 s`** on `macmon` `L2` windows, whereas sub-second operations must rely on `0 Hz` boundary `L3` `ri_energy_nj` counters or batched windows.

6.4 Falsifier 3 & Step 7: `B` temporal boundary vs. `C` proportional (`Kepler`) attribution

Across `3` replicates on `Apple M5 Max`, a single `SUT` executed **Phase 1** (`2.50 s` ALU-dense `SHA-256` digest burst: `29.27 ms` mean `CPU` time, `38.63 mJ` per `CPU`-ms) followed sequentially by **Phase 2** (`2.51 s` `192 MiB DRAM` cache-miss sweep + synchronous `fsync` disk flushes and `8 ms I/O` waits: `15.94 ms` mean `CPU` time, `27.73 mJ` per `CPU`-ms) alongside a steady duty-cycled background worker:

Attribution check (`n = 3` sequential two-phase + co-tenant replicates)	Phase 1 (`Compute 2.50 s`)	Phase 2 (`DRAM + fsync I/O 2.51 s`)	Falsifier / Step-7 verdict
Exact `B` temporal boundary `L3` delta (`reported_j`)	`1.1272 J (1,127,172 µJ)`	`0.4433 J (443,317 µJ)`	**`Step 7` residual `<= 1 µJ` (`3/3` pass)**
`C` (`Kepler` `ΔCPU` share: `64.58%` vs. `35.42%`) on `L3`	`1.0145 J (-10.00%)`	`0.5560 J (+25.21%` **mean err**`)`	`DRAM`/stall energy/ms ≠ `ALU` energy/ms
Exact `B` temporal `L2` whole-`SoC` window (`reported_j`)	`82.246 J`	`90.975 J`	`DRAM`, `NVMe`, & host draw persist in `I/O` wait
`C` (`Kepler` `ΔCPU` share) applied to `L2` window	`111.758 J (+35.86%` **err**`)`	`61.464 J (-32.59%` **err**`)`	**`Falsifier 3` triggered (`> 20%` error)**

While `Step 7`'s sum check held to `0-1 µJ` (`100.0%` of trials) under boundary counter differences (`B`), `CPU`-time proportional attribution (`C`) over-credited the compute burst by **`+35.86%`** and under-credited the memory-and-`fsync` phase by **`-32.59%`** on shared `L2` (and erred by **`25.21%`** even on per-process `L3`), empirically confirming why `SATB` restricts `C` to overlapping parallel harnesses (`Shape H`) and flags **`> 20%`** sequential divergence via `Falsifier 3`.

6.5 Multi-session Falsifier 6 audit (188 windows) and Falsifier 8 harness-floor check

Auditing **188 real L2 IOReport windows** collected across three experimental campaigns on the shared Apple M5 Max host plus two paired Lane B vs. Shape H (Claude Code 2.1.330) evaluations confirms both session-level safeguards:

1. **Falsifier 6 baseline-drift audit (188 windows)**: Across exp-001 (11 windows: 1 within 5.0%, 10 drifted up to 44.5%), exp-002 + multi-turn (7 windows: 2 within 5.0% — dnull 1,190-instance batch at 3.09% drift and BigCodeBench Qwen 3.6 at 1.02% drift — and 5 drifted 12.16%–65.11% under concurrent yosys-abc synthesis), and ai-reads-human (170 windows at loadavg_1m = 12.92: 61 within 5.0%, 109 drifted), **64 / 188 windows (34.04%)** retained valid derived_marginal_j while **124 / 188 (65.96%)** automatically invalidated net-above-idle subtraction without losing a single raw L2 or per-PID L3 reported_j reading (188 / 188 = 100.0% raw survival).
2. **Falsifier 8 (Lane B vs. Shape H Claude Code 2.1.330)**: Across 24 paired /work formal and SQLite policy bundles, Lane B achieved **24 / 24 (100.0%)** on both Claude Opus 5.5 (44 model calls, 63,568 prompt+cache tokens, \$0.5477) and Claude Sonnet 5.5 (43 calls, 60,284 tokens, \$0.3413), matching Shape H (Claude Code 2.1.330: **24 / 24**, 89 calls / 218,449 tokens on Opus 5.5; 83 calls / 196,128 tokens on Sonnet 5.5) while reducing prompt+cache token traffic by **3.44× and 3.25×**. On 6 unseen multi-file Python bug-repair repositories (config-defaults, interval-merge, pagination, precedence, rate-window, retry-backoff), Lane B (Opus 5.5 28 calls / 52,045 tokens; Sonnet 5.5 31 calls; local Qwen 3.6 35B-A3B 42 calls / 2,342.05 J L2) again matched Shape H (Claude Code 2.1.330 + Opus 5.5: **6 / 6**, 30 calls / 86,744 tokens, 1.67× more tokens), confirming that Lane B is **not** a crippled harness floor.

6.6 Multi-platform hardware attribution across five silicon and OS archetypes

To verify that SATB conformance, Shape D fixed-point determinism, rootless PLATYPUS / HVCI permission boundaries, and physical shunt or cumulative NVML attribution hold across macOS, Linux, and Windows 11, we executed run_multi_host_satb.py (multi_host_satb_receipts.json) and run_windows_suite.py (windows_satb_receipts.json) across four additional Linux and Windows 11 hardware archetypes alongside Apple M5 Max:

Hardware archetype & OS kernel	Rootless sensor boundary & evidence class	Shape D 1M DNULLMM1 (10k probes)	C9 replay & C10 tree-kill mean	Single-read latency & 100 Hz achieved	Step 7 residual & Falsifier 3 (C vs. B)
Apple M5 Max (arm64, 128 GB, macOS 27.2.1)	IOReport L1/L2 + rusage L3 (reported_j; measured_j = None)	15027705736 (388.43 ns C++, 582.57 ns Rust)	149,747 B (0 diffs) · 2.073 ms (0 PIDs)	~120 ms IOReport · 7.44 Hz	<= 1 µJ · -32.59% Phase 2 (+35.86% Ph. 1)
NVIDIA Jetson AGX Thor (aarch64, 122.9 GiB, Linux 6.8.12-1021-tegra)	TI INA238 (hwmon6 VIN L1) + INA3221 (hwmon5 ch1–ch3 L2/L1, 0444 measured_j)	15027705736 (984.86 ns C -O3, 2,489.4 ns Py)	149,747 B (0 diffs) · 9.544 ms (0 PIDs)	2,948.8 µs I2C · 66.55 Hz	0.0 µJ · -70.01% ch2 (-67.02% VIN)
Framework AMD Ryzen AI Max+ 395 (x86-64, 122.8 GiB, Linux 7.0.0-43)	RAPL package-0 & core (0440 root:energy reported_j) + amdgpu PPT (0444; measured_j = None)	15027705736 (376.61 ns C -O3, 1,341.7 ns Py)	149,747 B (0 diffs) · 0.868 ms (0 PIDs)	24.34 µs sysfs · 94.28 Hz	0 µJ · -71.61% pkg-0 (-63.34% core)
Intel Core i9-13900H (x86-64, 14.7 GiB, Linux 7.0.0-43)	RAPL psys (L1) + package-0, core, uncore (L2, 0444 reported_j; measured_j = None)	15027705736 (420.93 ns C -O3, 863.6 ns Py)	149,747 B (0 diffs) · 1.120 ms (0 PIDs)	45.37 µs sysfs · 97.12 Hz	0 µJ · -72.89% pkg-0 (-73.62% psys)
Intel Core i7-1165G7 + RTX 2060 (x86-64, 63.7 GiB, Windows 11 26300)	nvml.dll RTX 2060 cumulative mJ (L2 reported_j) + Win32 L3; CPU RAPL = N (measured_j = None)	15027705736 (1,474.22 ns MSVC /O2, 2,885.6 ns Py)	149,747 B (0 CRLF) · 2.458 ms JobObj (0 PIDs)	1,174.2 µs NVML · 64.64 Hz (NT tick)	0 mJ · -89.88% time (-77.05% RDTSC)

Five cross-platform hardware and operating-system findings emerge from these live receipts:

1. **Bit-exact Shape D and Lane B replay invariance across macOS, Linux, and Windows 11 (arm64/aarch64 and x86-64)**: All five hosts — including Win32 CreateFileMappingA + MapViewOfFile compiled under MSVC 19.44 /O2 and NTFS file I/O on Windows 11 (NT 10.0.26300) — produced the exact 10,000-probe DNULLMM1 checksum **15027705736**, the exact test.setpoint frozen digest 0da4df8d9ccbce1e...4485781, and the exact 149,747-byte C9 transcript (SHA-256 d697ed1476129939... with **0 NTFS CRLF byte drift**), while reaping 4-PID runaway process trees in 0.868 ms to 9.544 ms via kernel primitives (SIGKILL on macOS/Linux and Win32 TerminateJobObject at **2.458 ms mean** on Windows 11, **50.0× faster** than spawning taskkill.exe /F /T /PID at 122.817 ms mean, and > 200× under the 2,000 ms specification ceiling).

2. **Physical INA238 / INA3221 shunt separation and Falsifier 6 under live GPU co-tenancy on NVIDIA Jetson AGX Thor**: On Linux 6.8.12-1021-tegra, both hwmon6 (ti,ina238 at i2c@c600000/ina238@44, VIN whole-board DC input at ~27.97 V) and all three channels of hwmon5 (ina3221@40: ch1 VDD_GPU, ch2 VDD_CPU_SOC_MSS, and ch3 VIN_SYS_5V0) are world-readable (0444) without sudo. During our strictly CPU-only probe (dedicated_host = false), an independent co-tenant llama-server held the host GPU lock:

Jetson AGX Thor physical shunt channel (0444 rootless sysfs)	Boundary	Pre-baseline (W)	Phase 1 Compute (measured_j / W)	Phase 2 DRAM+fsync (measured_j / W)	Post-baseline (W)	Drift (F6) & Kepler C Phase-2 err
hwmon6 INA238 VIN (module + carrier input)	L1	100.70 W	122.12 J (101.66 W)	99.02 J (81.94 W)	97.21 W	3.47% (A2 withheld: co-tenant) · -67.02%
hwmon5 INA3221 ch1 VDD_GPU (co-tenant LLM)	L2	29.26 W	40.29 J (33.54 W)	26.00 J (21.52 W)	28.13 W	External GPU co-tenant (-36.1% swing in F5)
hwmon5 INA3221 ch2 VDD_CPU_SOC_MSS	L2	36.68 W	45.73 J (38.07 W)	44.37 J (36.71 W)	36.08 W	1.64% (CPU/SoC/MSS domain) · -70.01%
hwmon5 INA3221 ch3 VIN_SYS_5V0 (5.08 V rail)	L1	14.89 W	18.66 J (15.53 W)	16.82 J (13.92 W)	13.98 W	6.13% > 5.0% (F6 triggered) · -68.85%

Across the four Falsifier 5 cadence windows, fluctuating co-tenant LLM requests shifted ch1 VDD_GPU from 35.36 J down to 22.60 J (-36.1%), pulling whole-board INA238 VIN (L1) from 107.48 J down to 81.73 J (-24.0%) even while ch2 VDD_CPU_SOC_MSS remained within 35.76-39.31 J. Because dedicated_host = false (and ch3 VIN_SYS_5V0 exceeded 5.0% drift at 6.13%), SATB withholds A2 whole-SUT net placement claims while preserving exact rail-separated measured_j (L1 and L2 channels never summed).

3. **PLATYPUS 0440 root:energy isolation and Strix Halo APU domain verification on Framework AMD**: On Framework Desktop (AMD Ryzen AI Max+ 395 w/ Radeon 8060S, Linux 7.0.0-43-generic), udev rule 90-energy-read.rules sets /sys/class/powercap/intel-rapl:0/energy_uj (package-0) and intel-rapl:0:0/energy_uj (core) to mode 0440 (root:energy, world_readable = false): ungrouped /work sandbox UIDs receive EACCES (blocking PLATYPUS side-channels), while the supervisor GID (energy) reads cumulative microjoules rootlessly at 24.34 µs median latency (0 µJ Step 7 residual). Inspecting hwmon9 (amdgpu) further resolved an open specification detail: Strix Halo's integrated Radeon 8060S exposes world-readable (0444) PPT power (power1_input / power1_average in µW, 75.38 W Phase 1 / 69.72 W Phase 2) rather than a separate energy1_input counter, whereas cros_ec (hwmon8) exposes fans and board thermals only.
4. **Bare-metal Windows 11 rootless boundary (0x20000 Basic User NVML vs. ring-0 RAPL block) and 15.625 ms NT clock-tick aliasing**: Under an explicitly de-elevated Basic User token (runas.exe /trustlevel:0x20000, IsUserAnAdmin() == false) on Windows 11 Pro (NT 10.0.26300, Intel Core i7-1165G7 + NVIDIA GeForce RTX 2060), in-process FFI calls to nvml.dll nvmlDeviceGetTotalEnergyConsumption (mJ) and nvmlDeviceGetPowerUsage (mW) succeed rootlessly in **1,174.15 µs median** (**37.4× faster** than spawning nvidia-smi.exe at 43.88 ms per call), clearing Step 7 with **0 mJ residual** (8.769 J + 8.816 J = 17.585 J, 3.0% pre/post drift). By contrast, unprivileged user-mode reads of Intel RAPL MSR 0x611 are blocked by the Windows NT kernel without a signed ring-0 driver (barred by OQ-5 and HVCI), and WMI EMI (\\.\EnergyMeter / Win32_PerfRawData_Counters_PowerMeter) exposes 0 instances, placing Windows CPU package L2 strictly in Label N ("not_available"). Furthermore, because unadjusted Windows NT updates GetProcessTimes on ~15.625 ms clock interrupts, 8 ms sleep-plus-FlushFileBuffers bursts in Phase 2 were aliased down to 62.5 ms (5.06% CPU share vs. 1,171.9 ms in Phase 1), causing Kepler C attribution to err by **-89.88%** (-77.05% even when using sub-tick RDTSC QueryProcessCycleTime cycles at 11.48% share).
5. **Sensor bus and OS timer cadence physics across all five platforms**: Whereas Linux x86-64 powercap sysfs reads complete in 24.34-45.37 µs (94.28-97.12 Hz achieved at 100 Hz), Windows NT's default 15.625 ms sleep quantum caps unadjusted 10 ms NVML loops at **64.64 Hz** while delivering **9.45 Hz** cleanly at SATB's 100 ms (10 Hz) reference tick, matching Jetson AGX Thor's 66.55 Hz I2C shunt ceiling (9.49 Hz at 100 ms ref) and contrasting with macmon's 7.44 Hz IOReport ceiling. Across **5 / 5 silicon and OS architectures**, CPU-proportional splitting (C) under-credited the 192 MiB DRAM + disk-sync phase by **32.59% to 89.88%**, confirming universally why SATB restricts C to overlapping parallel runs.

---

7. Open Hardware Verification and Limitations

Session falsifier	Consequence when triggered	Empirical status in this study
F1 Replay mismatch (C9)	Reference harness is non-deterministic	**Cleared on 5/5 hosts (0 byte/CRLF diffs across 149,747 B, 14 turns)**
F2 Sum check fails on > 5% of attempts	Attribution split broken on host	**Cleared on 5/5 hosts (0 µJ Linux; 0 mJ Win11 NVML; <= 1 µJ M5 Max)**
F3 B and C disagree by > 20% sequentially	C (Kepler) marked unreliable on host	**Triggered on 5/5 hosts (32.59%-89.88% error); C restricted**
F4 Known pulse load outside sensor accuracy	Sensor downgraded for session	**Open (L0 smart-plug sync-pulse rig not yet attached)**
F5 10 ms vs. 1 s tick shifts window energy	Telemetry observer effect	**Cleared (0.76% M5 Max L2 delta; 9.45-10.48 Hz achieved at 100 ms ref)**
F6 Pre/post resident baselines drift > 5.0%	Derived net channels invalidated	**Active (124/188 M5 Max windows + Linux co-tenant drifts cleanly gated)**
F7 Adapter passes C1–C12, fails canary	Conformance insufficient; session void	**Verified in simulation (session_void = true on swap)**
F8 Lane B places below strong Lane O harnesses	Lane B reported strictly as a floor	**Cleared (24/24 & 6/6 parity vs. Claude Code 2.1.330)**

Three concrete hardware and adapter items remain open (OQ-5):

1. **Dedicated quiesced A2 sessions and L0 AC wall-plug calibration**: Although physical TI INA238 (L1) and INA3221 (L2/L1) shunt measured_j rails are now verified live on NVIDIA Jetson AGX Thor alongside rootless reported_j counters on Apple M5 Max, Framework AMD Strix Halo, Intel Raptor Lake, and Windows 11 (RTX 2060 NVML), all five bench machines ran shared services or co-tenant workloads (dedicated_host = false), withholding A2 whole-SUT placement claims; attaching a sync-pulsed L0 AC meter (Shelly / Yokogawa) on a quiesced dedicated host is next.
2. **Integrated-GPU NVML counter check on Jetson AGX Thor**: While discrete-GPU nvmlDeviceGetTotalEnergyConsumption (mJ) is now verified rootlessly via nvml.dll on NVIDIA GeForce RTX 2060 (Windows 11), probing NVML cumulative energy on Jetson AGX Thor's unified Tegra GPU alongside its verified 0444 INA3221 ch1 VDD_GPU shunt awaits an idle window when the shared GPU lock is free.
3. **Broader Shape H and Shape I live roster**: Live external-harness runs in this study exercised Shape H on Claude Code 2.1.330 alongside Lane B and Shape D; executing live C1-C12 certification on Codex CLI, OpenHands, Terminus-2, and Shape I Inspect solvers is next.

References

1. Anthropic (2026). Claude Code sandboxing and MCP configuration documentation. https://docs.anthropic.com/en/docs/claude-code/security
2. ARC Prize Foundation (2026). ARC-AGI-3 interactive benchmarking and adapter harness specification. https://github.com/arcprize/
3. Chung, J.-W., Gu, Y., Jang, I., Meng, L., Bansal, N., & Chowdhury, M. (2024). Zeus / ML.ENERGY: Understanding and optimizing GPU energy consumption of DNN training and inference. *NSDI 2023 / ML.ENERGY*. https://ml.energy/
4. Cloud Native Computing Foundation (2025). Kepler (Kubernetes-based Efficient Power Level Exporter): Proportional CPU/GPU power attribution model. https://github.com/sustainable-computing-io/kepler
5. gerganov, g., & contributors (2026). `llama.cpp` HTTP server and prompt-cache determinism documentation. https://github.com/ggml-org/llama.cpp
6. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., et al. (2023/2026). Efficient memory management for large language model serving with PagedAttention and vLLM batch invariance notes. *SOSP 2023 / vLLM Docs*. https://docs.vllm.ai/
7. Lipp, M., Kogler, A., Oswald, D., Schwarz, M., Easdon, C., Canella, C., & Gruss, D. (2021). PLATYPUS: Software-based power side-channel attacks on x86. *IEEE Symposium on Security and Privacy (SP)*, 355–371. https://platypusattack.com/
8. Luccioni, A. S., Jernite, Y., & Strubell, E. (2024/2026). Power hungry processing and comparative disclosures of commercial AI energy figures. *FAccT / Hugging Face Blog*. https://huggingface.co/blog

9. Model Context Protocol (2025). MCP specification: Tools (`tools/list`, `tools/call`, and `tools/list_changed`). https://modelcontextprotocol.io/specification
10. NVIDIA Corporation (2025/2026). Jetson Thor series power management, INA238 carrier, and INA3221 rail monitoring guide. https://docs.nvidia.com/jetson/
11. Open Interface Engineering (2026). Metered Commit Gates; Task Satiation vs. Aggregate Rebound; Deterministic Null [same house]. https://research.openie.dev/
12. OpenAI (2025/2026). Codex CLI Linux Landlock/seccomp and macOS Seatbelt sandboxing architecture. https://github.com/openai/codex
13. Rundgren, A., Jordan, B., & Erdtman, S. (2020). JSON Canonicalization Scheme (JCS). *IETF RFC 8785*. https://www.rfc-editor.org/rfc/rfc8785
14. Terminal-Bench Team (2025/2026). Terminal-Bench 2 and Terminus-2 (Harbor) terminal agent evaluation benchmark. *arXiv / GitHub*. https://github.com/laude-institute/terminal-bench
15. UK AI Safety Institute (2025/2026). Inspect Evals: ReAct solver (`react()`) and Inspect Agent Bridge documentation. https://inspect.aisi.org.uk/
16. vladkens (2025/2026). `macmon`: Sudoless Apple Silicon power and performance monitoring CLI over `IOReport`. https://github.com/vladkens/macmon
17. Wang, X., Li, B., Song, Y., Xu, F. F., Tang, X., Zhuge, M., et al. (2025). OpenHands: An open platform for AI software developers as generalist agents. *ICLR 2025*.
18. Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., et al. (2024). OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *NeurIPS 2024*.
19. Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., & Press, O. (2024/2025). SWE-agent: Agent-computer interfaces enable automated software engineering, and `mini-swe-agent` reference implementation. *NeurIPS 2024 / GitHub*. https://github.com/SWE-agent/mini-swe-agent
20. Yang, Z., Adamek, K., & Armour, W. (2024). Accurate and convenient energy measurements for GPUs: Detailed study of `nvidia-smi` part-time power sampling. *arXiv:2312.12741*.