Technical Appendix: Adaptive Anticipatory Policy Trees

Anonymous submission

A Protocol details

Stage ladder. The study ran as a pre-registered ladder of stages, each with a written plan (goals, windows, seeds, endpoints, decision rules) committed before execution: pilot and planner diagnostics; guided-decoding adoption and the below-floor hypothesis test; the contested-window scan; hardening (planner retry, observer token sweep); the declared-primary confirmation; mechanism isolation (R1/P1/O1); external transfer; the $k \times n$ ablation; and replication (fresh-seed repeats, the four-model boundary map, and the two-model pre-registered gate confirmation).

Windows. Relaxed and floor anchors: 250, 500, 1000, 2000 ms. Contested-regime windows by stage: scan 600/650/700/750 ms (700 pre-registered primary, n.s.); confirmation 650 ms declared primary with 600/625/675 secondaries; mechanism isolation 650/600 ms; ablation 600 ms primary (recalibrated from 650 by the reactive-only protocol) with 575/650 secondaries; fresh-seed repeats 550 ms (sub-cliff); cross-model blocks at each model's own recalibrated knee (600 ms for UI-Venus and the confirmation generalist; 4–10 s scans for the slow dense models).

Paired-block anatomy. 42 pairs per block, key-balanced (14 per outcome), condition order alternating per pair, identical seed-derived hidden schedule within a pair, resume-safe manifests, zero tolerated pair-integrity failures. Success, incorrect-action, failure stage, and all latencies are recorded per trial; exact two-sided McNemar on discordant pairs is the only significance test used.

Depth rule for replication blocks. Fixed in advance: if a model's primary paired block shows the positive direction, run two repeat blocks (42 pairs each) at the same window; a null or negative primary gets one block plus a mechanism read. Boundary models that fail a capability gate persistently are stopped at the gate and reported (a paired block against a structurally failing arm can only reproduce 0/42).

Prompt-adaptation policy. Per model, at most one documented round of minimal prompt adaptation is permitted to meet the planner-validity gate (for grounding-specialized models this is typically a hardened flat-children instruction); the adapted prompt is diffed, frozen, and used for all subsequent blocks of that model. Base-prompt behavior is always reported alongside. The planner retry-on-invalid is bounded (one resample at unchanged temperature) and its firing is recorded per trial.

Seed ledger. Seeds are integers whose hidden schedules derive deterministically; every consumed range is burned and never reused. Burned enclosing ranges: 14001–14005 and 40001–40020 (pilot), 48001–48006, 49001–49003, 50001–50062, 51001–51241 (diagnostics), 52001–52075 (guided-decoding arms), 53001–56047 (window sweep), 57001–60047 (contested scan), 61001–61034 and 62001–62382 (hardening), 63001–63202 and 64001–64018 (confirmation and click-target probe), 65001–65099 (mechanism isolation), 66001–66112 and 66901–66903 (ablation), 67001–67084 (fresh-seed repeats), 68004–68019, 68100–68161, 68300–68314 (boundary models), 68400–68591 and 68900–68939 (gate confirmation). The external benchmark uses its own fixed task set (39 tasks, upstream commit pinned) and burns no local seeds.

B Policy-tree JSON Schema

The guided-decoding schema for the planner (the $k \times n$ ablation generates per-cell variants of the same shape with adjusted bounds). Note the flat node list with string-id `children`; the recursion-free shape is itself load-bearing (Section 5.4). For the three-outcome benchmark event, the canonical compiled tree this schema yields is:

```
root: idle screen -> WAIT
 |- guard "Press F12"   -> press('f12')
 |- guard "Press Enter" -> press('enter')
 |- guard "Press Esc"   -> press('esc')
```

```
{
  "type": "object",
  "required": ["reason", "cover_ms", "
    root_id", "nodes"],
  "properties": {
    "reason":   {"type": "string", "
    minLength": 1},
    "cover_ms": {"type": "integer", "minimum
    ": 1},
    "root_id":  {"type": "string", "
    minLength": 1},
    "nodes": {
```

```
      "type": "array", "minItems": 1, "
    maxItems": 10,
      "items": {
        "type": "object",
        "required": ["id", "depth", "guard",
     "action", "confidence",
                     "deadline_ms", "risk",
    "expected_state", "children"],
        "properties": {
          "id":              {"type": "string
    ", "minLength": 1},
          "depth":           {"type": "
    integer", "minimum": 0, "maximum": 2},
          "guard":           {"type": "string
    ", "minLength": 1},
          "action":          {"type": "string
    ", "minLength": 1},
          "confidence":      {"type": "number
    ", "minimum": 0, "maximum": 1},
          "deadline_ms":     {"type": "
    integer", "minimum": 1},
          "risk":            {"enum": ["low",
     "medium", "high"]},
          "expected_state": {"type": "string
    ", "minLength": 1},
          "children": {
            "type": "array", "maxItems": 3,
            "items": {"type": "string", "
    minLength": 1}
          }
        },
        "additionalProperties": false
      }
    }
  },
  "additionalProperties": false
}
```

Validation beyond the grammar: executable-action grammar and allowlist, `cover_ms` against the measured planner budget, guard presence on every executable branch, child-id resolution and reachability, and node/depth/branch budgets. (The fallback path is a runtime property – observer abstention – not a validated tree feature; see Section 3.6.) The grammar backend does not enforce `maxItems`, which is exactly the gap the validator plus bounded retry closes (Section 3.5).

C Observer prompt and a routed example

The complete fast-observer prompt template (`fast_observer.txt`), mirroring the tree-schema listing above:

```
You are a fast visual branch selector.
    Inspect the
screenshot and the eligible branch guards.

Return exactly one JSON object: {"branch_id
    ":"<id or
NONE>","confidence":0.0,"reason":"brief
    visual evidence"}.

Select a branch only when its guard is
    visibly satisfied.
Do not generate actions, future plans,
    markdown, or extra
text. If the screenshot is ambiguous,
    unchanged, or matches
no guard, return NONE. Be concise.
```

A real routed decision (B.1, seed 63010). Eligible guards passed with the gated frame (the compiled tree's three children; deadlines in ms):

```
[{"id": "prompt_f12",   "guard": "Prompt
    requests F12",   "deadline_ms": 500},
 {"id": "prompt_enter", "guard": "Prompt
    requests Enter", "deadline_ms": 500},
 {"id": "prompt_esc",   "guard": "Prompt
    requests Esc",   "deadline_ms": 500}]
```

Observer response (24 completion tokens, 246.9 ms):

```
{"branch_id":"prompt_enter","confidence
    ":1.0,
 "reason":"The prompt explicitly requests
    pressing Enter."}
```

The runtime then checked eligibility, the planner-side branch confidence ($0.95 \geq \tau_m = 0.45$), and decision age (249.3 ms $\leq$ 500 ms deadline) before executing the pre-authorized `pyautogui.press('enter')`. The preceding static frame had returned `NONE` (the abstain path exercised in the same trial).

D Per-model serving provenance

All models served with vLLM 0.25.0, BF16, pinned Hugging Face revisions, `max_model_len` 8192, on RTX 6000 Ada GPUs (TP=2 except the 7B at TP=1). The FlashInfer sampler is disabled by launch environment (its JIT is incompatible with the installed CUDA headers *and* the env var is part of the compile-cache key). Thinking modes are disabled where the chat template honors it; one model required a no-think template override (pre-filled empty think block), documented as its adaptation.

Reproducibility note (serving numerics). The compile-cache key is part of the experimental condition. In the planner diagnostic stage, adding `--revision` and an offline flag to an otherwise identical launch changed the cache key, forcing fresh kernel autotuning; tree validity fell from 88.1% to 18.3% on identical weights, sampler, prompts, and pixel-identical screenshots, and recovered exactly when the original byte-identical launch (and thus the original compiled kernels) was restored. The failure flavor was a single token-level mode flip (the root node's `"action": "WAIT"` field present vs. omitted). Consequences adopted study-wide: byte-exact launch lines are part of provenance; the cache key is read from the server log and verified before data collection; per-model keys are frozen (Table S1); and any key change voids collected data.

Token-normalized costs. Each model's decode rate is fit from a forced-length token bench in both prompt regimes (planner-sized and observer-sized inputs; $R^2 \geq 0.99$), giving tok/s and a fixed per-call overhead. Critical-path costs in the paper convert measured wall-clock latencies through

Table S1: Frozen serving identities. The compile-cache key is vLLM's `torch.compile` cache identity, verified in the server log before every data-collecting run; revisions are abbreviated to eight hex digits.

Model	HF revision	Compile-cache key	Launch
Holo-3.1-35B-A3B	`2bdb9285`	`cca9e728f1`	TP=2
Qwen3.6-35B-A3B	`995ad96e`	`2e9fc7f7e2`	TP=2
UI-Venus-1.5-30B-A3B	`b6e8f31f`	`8fc4b0969f`	TP=2
EvoCUA-32B	`ce055343`	`0f5d6516ae`	TP=2, no-think template
UI-Ins-32B	`9ee51926`	`f1cbc332a7`	TP=2, forced bf16
UI-TARS-1.5-7B	`683d002d`	`ff3c9ea5a8`	TP=1

the serving-stack rate (e.g., 181 tok/s for the primary model) into output-token equivalents, so a reader can rescale every window and latency to their own hardware.

E Round-1 addition: five-outcome event family (EXP-A)

To probe the coverage rule of Section 3.4 beyond the degenerate three-outcome case, we added a five-outcome benchmark scenario (uniform over F12/Enter/Esc/F2/F9, seed-derived and byte-stable against the historical three-key schedule) and pre-registered endpoints, seeds, and predictions before data collection.

Validity gate (planner-side). With $K_{\max} = 5$ and no prompt adaptation, tree validity was 29/30 and 29/30 trees covered all five outcomes: the planner ports to the larger outcome family unchanged.

Window calibration and a null primary result. The pre-registered contested-window scan $\{500–800\,\text{ms}\}$ floored R0 (the live reactive baseline) at 0/10 on the five-outcome event, while a concurrent sanity check on the three-key event reproduced historical behavior — the larger outcome set alone slows the reactive model's decisive call by ≈150–200 ms, shifting the contested band right. Under the pre-declared widen-once rule we scanned $\{850–1000\,\text{ms}\}$, declared 900 ms (scan R0 $= 0.50$) before any treatment trial, and ran the primary 42-pair block. The result is null: R0 went 42/42 (ceiling) while T2 went 38/42 (0.905); exact McNemar $p = 0.125$ with all four discordant pairs favoring R0. The mechanism is serving non-stationarity: R0's reaction distribution shifted ≈100 ms faster between scan and block (block p50/p95 $= 730/810.6$ ms) on the same server process and verified frozen compile-cache key, so the window that was contested at scan time had ceilinged by block time. We report the null verbatim rather than moving the window post hoc; it is itself evidence for the paper's core caution that ∼100-ms-scale serving drift changes outcomes at contested windows. Planner-side gates at 900 ms were unaffected: validity 0.952, outcome recall 0.929, visible-guard branch accuracy 0.974, zero incorrect actions from T2.

Exploratory contested block. Because the declared window ceilinged, we ran one additional 42-pair block at 750 ms – chosen from the primary block's measured R0 reaction distribution (p50 $=$ 730 ms), declared before any 750 ms trial, and labeled exploratory since its window was not set by the pre-registered scan procedure. There the regime is contested and the paper's mechanism reproduces on the five-outcome family: T2 $= 25/42$ (0.595) vs. R0 $= 11/42$ (0.262), discordant pairs 16/2 in T2's favor, exact McNemar $p = 1.3 \times 10^{-3}$. T2's decline from 0.905 at 900 ms is late-routing pressure (observer decode ≈283 ms against the tighter window), the same dominant miss mode as the main-benchmark taxonomy (Section 5).

Table S2: Five-outcome family, branch-budget cells (42 pairs each, 900 ms). The pre-registered knee at $k =$ outcome-set size $= 5$ is confirmed; the k=2 validity valley of Section 5.6 reappears at k=3.

$K_{\max}$	Validity	T2 success	Dominant failure
1	24/42	2/42 (0.048)	schema, misroute
2	37/42	12/42 (0.286)	misroute, routing
3	**2/42**	1/42 (0.024)	budget refusal
5	33/42	**32/42 (0.762)**	schema
7	39/42	19/42 (0.452)	armed catch-all

Branch-budget ablation: the knee moves to $k = 5$. The pre-registered prediction was that the knee of Section 5.6 tracks the outcome-set size. It does (Table S2): T2 success peaks at $K_{\max} = 5$ and is degraded on *both* sides, so the optimum is interior, not a saturation.

The k=3 valley has an identified mechanism we call *budget refusal*: 38/42 failures are the post-hoc validator rejecting `children` for `maxItems: 3` because the planner, able to see all five prompt outcomes, emits all five children rather than truncating to its budget (the bounded retry then re-fails). This generalizes the k=2 uncanny valley observed on the three-outcome family (validity 0.405 there, 0.048 here): the valley sits at budgets just below the visible outcome-set size, where the planner will neither specialize (as at $k \leq 2$) nor cover (as at $k \geq 5$).

Mis-set budgets execute wrong actions. The budget cells bound the zero-incorrect-actions property of Section 3.6: wrong keys were actually pressed in 13/42 trials at k=1, 12/42 at k=2, and 15/42 at k=7 — versus 0/42 at the matched budget k=5. The two mechanisms are instructive. Under-budgeted planners ($k < 5$) write *generic* guards ("Prompt visible" rather than "Prompt requests F12"), which the observer matches on any outcome, firing the covered key. The over-budgeted planner (k=7) fills its surplus slots with catch-all branches ("Unknown prompt") that *carry a default action* (28/39 valid trees); the observer correctly routes am-

biguous frames to the catch-all, and the default action fires. Guard specificity and a passive fallback are planner behaviors, not validator-enforced invariants, so the branch budget is a correctness parameter: it should be set to the enumerable outcome-set size, and mis-setting it in either direction converts misses into wrong actions.

Uneven outcome probabilities (coverage rule, Eq. 3). A second arm reweights the five outcomes to $.35/.30/.20/.10/.05$ under a $K_{\max} = 4$ budget, so covering to $\rho = 0.95$ requires dropping exactly the rarest outcome (F9). The pre-registered planner-side prediction held: 30/30 trees valid, and 29/30 dropped exactly F9, keeping the four most likely branches; in the paired block (42 pairs, 900 ms) 39/42 compiled trees again dropped exactly F9.

The paired endpoint, reported verbatim, was zero: T2 $0/42$ against R0 $39/42$. The traces attribute all of it to a single design interaction we had not anticipated: told the outcome distribution, the planner copies the *priors* into each branch's `confidence` field (leaf confidences exactly $.35/.30/.20/.10$ across the block, versus guard-conditional 0.80–0.95 in the even family), and the runtime's suppression check (ii) – the planner-confidence floor $\tau_m = 0.45$ – then rejects every route. All 80 observer calls that selected a branch were suppressed as `low_confidence`, including plainly correct ones; replaying the rejected routes against ground truth shows 40/42 would have fired the right key (0 the wrong key) with decision ages $\leq$ 394 ms against a 500 ms branch deadline. The observer and the drop-rarest trees are sound; the floor alone zeroes the arm. The lesson mirrors the budget finding: Q_v does double duty as calibrated action confidence and, once priors are visible to the planner, as an outcome prior – and $P(\text{outcome})$ is not $P(\text{action correct} \mid \text{guard matched})$. Any outcome family whose maximum prior sits below τ_m is unroutable by construction. Separating the two fields (a branch `prior` distinct from `confidence`), or gating outcome-branch nodes on observer-match confidence instead, is the recorded fix direction; we report the null rather than patching and rerunning.

F Round-1 addition: prep-budget sweep and delay jitter (EXP-B)

What "prep" means operationally. Prep is the budget from task start (quiet screen visible) to event onset – the time the planner has to compile, validate, and arm a tree. The sweep uses a *prep-race* mode that arms the event countdown *before* planning starts, so the declared delay is the whole prep budget; in all other experiments the countdown is armed only after the tree is ready. The window was declared from a fresh reactive-only scan (700 ms, R0 $= 0.50$), and the planner's latency was stable across cells (p50 ≈ 1.8 s, p95 $\approx$ 1.9 s), so Eq. (2) predicts recovery at $L_{p95} + M \approx 2.4$ s.

The crossover brackets $L_{p95} + M$, and starved prep fails safe. With prep $\in \{0.5, 1\}$ s – below the planner's own p50 – T2 scores $0/42$ in both cells: the event fires mid-compile, no tree is armed, and every failure is a *miss*, never a wrong action (0 wrong executions across all 168 sweep T2 trials; the dominant failure stage is the change gate firing with no tree to route into). At prep $= 2$ s, just below the predicted threshold, T2 recovers to $36/42$ but shows no advantage over the reactive arm ($37/42$, discordant $0/1$, $p = 1$): the arming margin is thin and six trials miss at the benchmark deadline. At prep $= 4$ s, above the threshold, T2 lands $42/42$. The deployer-facing summary: AAPT needs its own measured $L_{p95} + M$ of quiet screen before the event; give it less and it degrades to the reactive baseline or below, but it degrades by *missing*, not by acting wrongly.

One caveat is reported verbatim: the reactive arm scored $0/42$ in the 4 s cell, and the trial payloads attribute this to *format drift*, not timing – after $\sim$8 consecutive idle polls the reactive model abandons the executable-string action format for a nested object that the executor rejects. T2's $42/42$ recovery stands on its own; the $42/0$ discordant split against R0 in that cell partially reflects this unrelated reactive fragility (long idle phases degrade reactive format adherence), which we flag as its own finding about reactive baselines rather than absorb into the AAPT comparison.

Delay jitter: the fixed-delay leak concern, answered with data. A paired block with the event delay drawn per seed from $U(500, 1000)$ ms (instead of the fixed 750 ms) leaves both arms' behavior consistent with the mechanism claim: T2 holds $36/42$ (Fisher vs. a fixed-delay reference block at the same window: $p = 0.11$, n.s.), and within the jitter block T2 beats R0 at $15/0$ discordant pairs ($p = 6.1 \times 10^{-5}$). Nothing in the runtime conditions on the delay value – routing is change-gate plus observer driven – and randomizing the delay does not remove the gain, so the fixed 750 ms delay of the main protocol is not an information leak. The reactive arm's cross-block Fisher is significant ($21/42$ jittered vs. $38/40$ fixed, $p = 4.2 \times 10^{-6}$), but the attribution is confounded: the same fixed-delay window swung from R0 $= 0.50$ (scan draw) to 0.95 (reference draw) across seed draws, and the prep sweep shows R0's decisive-call arrival is phase-locked to the delay value (success $37 \to 26 \to 37$ across fixed prep values). Both effects sit on the reactive side; the direction – randomization makes the *baseline* no better and leaves the treatment unchanged – is the opposite of what a timing leak in AAPT's favor would produce.

G Round-1 addition: oracle-routing probe on UI-Venus (EXP-D)

To test whether the routing gate (c) of Table 2 is causal rather than correlational, a pre-registered probe re-ran the UI-Venus-1.5-30B-A3B pairing with the live observer replaced by ground-truth routing (condition O1: zero-latency keystroke oracle, gated on the same change signal, able to fire only branches present in the compiled tree). Everything else was frozen – same one-round-adapted prompts, same 600 ms window as the original paired block, fresh seeds, serving compile key verified unchanged. Our committed prediction was that oracle routing would *not* beat R0, on the premise that the original block's tree coverage (outcome recall 0.22) caps the oracle below R0.

The prediction failed: O1 succeeded on 22/42 vs. R0 12/42 (discordant 13/3, exact McNemar $p = 0.021$), while R0 reproduced its original rate (12/42 vs. 13/42). The de-

composition is exact, with no residual: all 22 successes are precisely the trials whose tree was schema-valid *and* covered the true outcome; the oracle abstained on all 10 valid-but- uncovered trees (deadline miss, no wrong action) and 10 trials failed schema validation. O1 executed zero incorrect actions; R0 executed wrong actions in 30/42 trials. The prediction failed because its premise did: this session's unseeded planner samples had recall 0.59 (vs. 0.22 in the original block, identical config and compile key – a between-session distribution shift on a planner already documented as numerically knife-edged, Section 4.4). The licensed reading is therefore conditional and two-sided: routing quality is causally binding (perfect routing alone turns the tie into a significant win), and planner outcome-recall remains co-binding (the oracle lands exactly at its coverage ceiling, $22/42 =$ the covered fraction). The UI-Venus row of Table 2 is annotated accordingly.

H Round-1 addition: survey of documented response windows (EXP-E)

To situate the benchmark's contested band in deployed systems rather than constructed ones, we surveyed documented response windows – deadlines a user or agent must act within (*react*) or transient UI that is only visible, and hence only actionable, for a bounded time (*display*). Figure S1 places the 19 documented windows of Table S3 on a log axis against the paper's contested band and the measured reactive p50.

The population divides at the measured reactive latency. Below and inside the contested band – parry windows, double-tap/double-click/long-press timeouts, fastboot hotkey windows – a 567–730 ms reactive loop is structurally excluded or coin-flip, and pre-computation is the only mechanism that acts in time. In the 1–10 s band (toasts, snackbars, banners, GRUB, WALK intervals) a reactive VLM loop competes but with thin margins under load or multi-step responses, and the $\pm$100 ms serving non-stationarity we measured (Appendix E) is a meaningful fraction of the shortest of these. At or above 10 s (regulated takeover, WCAG, TOTP, push approval, SSH) the reactive loop suffices and AAPT's premium is not justified. The survey bounds the claim honestly in both directions: real sub-second windows exist in deployed systems, and most windows above a few seconds do not need anticipation.

I Comparative map of the closest prior systems

Table S4 maps the systems discussed in Section 2 feature by feature. AAPT's distinguishing column combination is live observation routing through a world-model-generated tree with explicit latency coverage and deadlines.

J Method details moved from the main text

J.1 Dataflow diagram

Figure S2 shows the complete dataflow of the implementation described in Section 3 of the main paper.

J.2 Branch allocation rule

The generation-time allocation rule for the per-node branching factor K_v of Section 3.4 is

$$K_v = \begin{cases} 1, & \text{confidence} \geq \tau_c \text{ and risk} \leq \tau_r, \\ \min(K_{\max}, m_v), & \text{outcome uncertain,} \\ 0, & \text{safe fallback / replan required,} \end{cases} \tag{1}$$

subject to global budgets: per-node limit $K_{\max}$, maximum depth $n_{\max}$, total node budget, coverage target T_{cover}, and risk-specific open-loop horizons. The hard budgets ($K_{\max}$, $n_{\max}$, node count, `cover_ms`) are enforced numerically by the post-hoc validator, whereas τ_c, τ_r, ρ, and m_v are *design rules realized as planner-prompt instructions*, not runtime-enforced thresholds – the planner is asked to branch on uncertain outcomes and cover the likely ones, and the rule describes the behavior this induces.

J.3 Engineering findings behind guided-decoding planning

Both findings were initially misdiagnosed as capability gaps. **Schema shape dominates grammar constraints.** With the original schema, which expressed "an action or an action array" as a JSON-Schema `oneOf` alternation, tree validity was 88.1% (238/270) and the dominant failure was a node omitting its required action field entirely. Removing the alternation (scalar `action` only) fixed validity to 60/60 *even without any decoding constraint*, while grammar-constrained decoding with the original schema reached only 50/60. The deficit was induced by the schema, not by the model; guided decoding plus the simplified schema yields 230/231 (99.57%) pooled validity, and a planner LoRA fine-tune that had been provisionally supported by the earlier numbers was cancelled. **Post-hoc validation plus bounded retry closes the residual gap.** The grammar backend does not enforce JSON-Schema array-length bounds (`maxItems`), so a rare over-wide or unreachable tree still decodes; a strict validator (action grammar and allowlist, deadline and coverage checks, guard presence on every executable branch, reachability, budget enforcement) rejects it, and the planner resamples once at unchanged temperature. In live operation the retry fired on 7/336 trials and recovered 4, leaving 0.89% final invalidity.

J.4 Change-gate constants and latency accounting

The pixel-difference gate that feeds the observer is fully specified by three constants, fixed across all experiments: a frame is "changed" when at least 0.5% of pixels (`min_changed_fraction` $=$ 0.005) differ from the previous gated frame by $\geq$ 20 grayscale levels (`pixel_threshold` $=$ 20, absolute difference on grayscale), and a static-scene probe frame is forced through every 500 ms (`static_probe_ms`) so a stalled screen cannot silence the observer indefinitely. Capture runs at 30 FPS, so the gate contributes at most one frame interval ($\leq$ 33 ms, expected $\approx$ 17 ms) between event onset and the frame the observer decodes; T2's reported reaction time includes

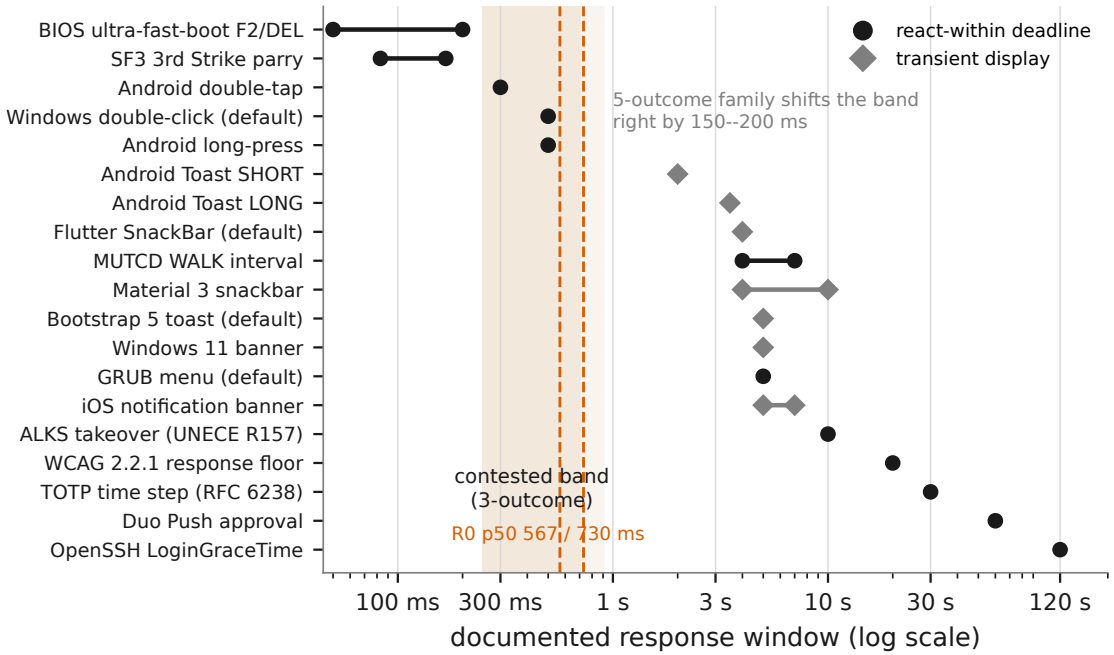

Figure S1: Documented real-world response windows (log scale) against the benchmark's contested band (shaded; the five-outcome family shifts it right by 150–200 ms) and the measured reactive p50 (dashed). Below the band a reactive loop is structurally excluded; inside it, contested; an order of magnitude above it, sufficient.

this capture quantization plus observer decode and routing checks.

J.5 Model inventory

Table S5 lists the six checkpoints with measured observer-regime decode speeds and their roles in the study.

J.6 Deliberately unclaimed extensions

Two natural extensions from the original design are not claimed: asynchronous tree refresh (overlapping planning windows, so a successor tree is compiled while the current one controls the fast path) and reusable policy memory (promotion of repeatedly validated subtrees into PreAct-style procedural skills, with promotion gates on repeated success and revocation support). Both remain design sketches.

K Sizing-rule details: grids, frontier, and token sweep

K.1 The $k \times n$ frontier

Six cells ($k \in \{1, 2, 3\}$, $n \in \{1, 2\}$, per-cell generated guided-decoding schemas), 42 pairs per cell against a shared reactive arm, paired by seed. The pre-registered 650 ms primary sat at R0 ceiling this session (0.952; serving was faster than in the confirmatory stage), so the window was recalibrated by the fixed reactive-only protocol to 600 ms (R0 $= 0.548$ on the full 42 primary seeds) – already a pre-registered secondary – with 650 ms retained as a ceiling anchor and 575 ms as a harder secondary. Costs are reported in milliseconds and in output-token equivalents at the measured 181 tok/s.

The ordering is fully window-robust: at the harder 575 ms secondary (R0 $= 0.071$), $k3\,n1$ still lands 42/42 while all $k < 3$ cells stay floored. k=1 stays more valid (0.857) than k=2 (0.405) but commits to a single outcome and is coverage-floored. On depth: the task has one decision step, and the planner builds the same four-node depth-1 tree even when 13 nodes and depth 2 are allowed (realized tree p50 $= 4$ in both k=3 cells); for $k < 3$ extra depth actively collapses validity further ($k1\,n2$: 0.119). The critical-path cost is nearly flat across all six cells (observer p50 287–325 ms $\approx$ 52–59 token-equivalents): accuracy is set by planner-side validity and coverage budget, not observer latency. Planner prep cost (off the critical path) grows from $\sim$1.0 s to $\sim$1.8 s with budget.

K.2 Observer token-cap sweep

The failure modes bracket the knee (Table S7): 32 tokens truncates the observer's JSON mid-object (pooled 0.571, 27 failures), while 128 tokens adds latency and drifts branch accuracy below the gate (0.829). Adopting the 64-token observer lifted the contested-regime T2 ceiling from $\sim$0.70 (split between wrong-branch routing and late second observer calls) to 0.79/0.88 at 600/650 ms, with zero late routings in all eight cells of the adopting run. The planner side was hardened in the same stage by the bounded retry-on-invalid: acceptance 30/30 valid, and over the 336 live trials the retry fired seven times and recovered four invalid trees,

Table S3: Survey of documented response windows (EXP-E). Sources are primary documentation; "configurable" notes whether deployments can lengthen the window.

System	Window	Config.	Class	Source
BIOS ultra-fast-boot F2/DEL	≈0–200 ms	fast-boot toggle	react	vendor fast-boot docs
SF3 3rd Strike parry	83–167 ms	no	react	wiki.supercombo.gg (5–10 frames @60 fps)
Android double-tap timeout	300 ms	no	react	developer.android.com `ViewConfiguration`
Windows double-click default	500 ms	yes (max 5 s)	react	learn.microsoft.com `GetDoubleClickTime`
Android long-press timeout	500 ms	accessibility	react	developer.android.com `ViewConfiguration`
Android `Toast.LENGTH_SHORT`	2 s	no	display	Android docs constant
Android `Toast.LENGTH_LONG`	3.5 s	no	display	Android docs constant
Flutter SnackBar default	4 s	yes	display	api.flutter.dev `SnackBar.duration`
MUTCD WALK interval	4–7 s	agency-set	react	mutcd.fhwa.dot.gov §4I.06
Material 3 snackbar guidance	4–10 s	yes	display	m3.material.io snackbar spec
Bootstrap 5 toast default	5 s	yes	display	getbootstrap.com toasts
Windows 11 notification banner	5 s	min 5 s	display	learn.microsoft.com
GRUB menu timeout default	5 s	yes	react	GNU GRUB manual `GRUB_TIMEOUT`
iOS temporary notification banner	5–7 s	no	display	discussions.apple.com
ALKS takeover request	10 s	regulated	react	UNECE R157 transition demand
WCAG 2.2.1 extend-session response	20 s	regulated floor	react	w3.org WCAG SC 2.2.1
TOTP time step	30 s	rarely	react	RFC 6238
Duo Push approval timeout	60 s	no	react	help.duo.com
OpenSSH `LoginGraceTime` default	120 s	yes	react	`sshd_config`

cutting final planner invalidity from 2.08% to 0.89%.

K.3 Window sensitivity and the recalibration protocol

The contested window is a property of the model-serving-seed triple, not a benchmark constant: it moved from 600–650 ms (shared-server sessions) down to ∼600 ms and below (dedicated server, faster decode), and across seed draws it can vanish entirely into a near-vertical cliff. Two protocol rules kept this honest: (i) windows are recalibrated only from reactive-only scans before any treatment data, and only toward windows that were pre-registered as secondaries; (ii) when a fresh seed set has no contested midpoint, the mechanism is tested at a sub-cliff window instead, where the paired design still discriminates (the reactive arm cannot reach the deadline that the ∼300 ms observer can). Per-seed reactive latency is stable across sessions (p50 567 ms, reproduced exactly an hour apart), so these shifts are difficulty variance across 42-seed draws, not server drift.

L Additional results detail

L.1 Prediction ledger: Round-1 revision entries

Table S8 continues the main paper's prediction ledger with the Round-1 revision entries.

L.2 Mechanism-isolation figure

Figure S4 compares all five conditions at the two contested windows (Test 2 of the main text).

L.3 Contested-window scan

Table S9 gives the full four-window scan behind Test 1's contested-regime localization.

L.4 Cost accounting and miss taxonomy (declared-primary block)

On the declared-primary block (Table S10), T2 generates $3.2\times$ R0's completion tokens per trial (342.8 vs. 105.8): the tree compilation (283.5 completion tokens, p50 1.81 s) is the premium, and the runtime observer is *cheaper* per call than R0's decisive calls (59.3 vs. 105.8). Every trial compiled exactly one tree and 33/42 triggered successfully (1.27 trees per successful trigger); the premium buys the off-critical-path position, worth paying exactly when the window is contested. T2's nine misses on the block: 5 routed-late-executed (the observer chose the right branch but its decision age, 382–446 ms, plus execution landed past the server deadline), 3 wrong-branch routes (all three chose the F12 branch – an observer bias – and all three presses also landed post-deadline, so they score as misses rather than incorrect actions), and 1 correctly suppressed late decision. On the same block every pre-registered secondary window is also individually significant (600 ms: 30/42 vs. 6/42, $p = 1.2\times10^{-7}$; 625 ms: 30/42 vs. 2/42, $p = 7.5\times10^{-9}$; 675 ms: 35/42 vs. 27/42, $p = 7.8\times10^{-3}$), with tree validity 1.0 (625 ms: 0.976).

L.5 Confirmation blocks on the untuned generalist

Figure S5 shows the three 42-pair confirmation blocks of Section 5.3 individually: every block is significant on its own, the direction never reverses, and the pooled 126 pairs give T2 0.778 vs. R0 0.341 (discordant 60/5, $p = 4.9\times10^{-13}$). Gate measurements on this model: decode 180–201 tok/s; tree validity 30/30 with full outcome coverage; branch accuracy 0.969 pooled, expected calibration error 0.020. Incorrect-action rate in the primary block: R0 0.57, T2 0.00.

Table S4: Comparative map of the closest prior systems. • = implemented and evaluated; (•) = partial (qualifier beneath); ○ = absent.

Work	Alternatives	Live routing	Model-free exec.	Deadlines
WebDreamer	(•) candidate lookahead	○	○	○
MobileDreamer	(•) prediction tree	○ advises only	○	○
TraceR1	○ one trajectory	○ replans each step	○	○
Speculative Actions	(•) top-k breadth	(•) slow-actor commit	(•) speculative fast path	(•) API latency
AOI	○	(•) captures, no routing	○	(•) perception only
PreAct	(•) from experience	• verify-then-replay	•	(•) repeat efficiency
AAPT	• contingency tree	• guard routing	• within tree	• coverage + deadlines

Table S5: Models studied (CUA = computer-use agent; MoE = mixture-of-experts). Decode is measured observer-regime speed on our serving stack (1–2 RTX 6000 Ada GPUs, tensor parallel 2 for the 30–35B models); "observer call" is the latency of one routing call. MoE models decode roughly 7× faster than the dense 32B models, one of three gates for the AAPT gain.

Model	Class	Decode (tok/s)	Observer call	Role in this paper
Holo-3.1-35B-A3B	Qwen3.5-MoE, CUA-tuned	∼181	∼0.2 s	primary model
Qwen3.6-35B-A3B	Qwen3.5-MoE, generalist	180–201	∼0.15–0.45 s	confirmation model
UI-Venus-1.5-30B-A3B	Qwen3VL-MoE, CUA-tuned	∼170	∼0.15 s	gate (c) boundary
EvoCUA-32B	Qwen3VL dense, CUA-tuned	∼25	∼2.5 s	gate (a) boundary
UI-Ins-32B	Qwen2.5-VL dense, CUA-tuned	∼26	∼1.5 s	gates (a)+(b) boundary
UI-TARS-1.5-7B	Qwen2.5-VL dense, CUA-tuned	∼59	∼0.5–1.1 s	gate (b) boundary

L.6 DynaCU-Bench category table

Table S11 gives the per-category outcomes behind the Test-5 dissociation. Episode-level behavior is near-deterministic (5 of 78 task-condition cells non-unanimous). Median decision latency, measured in the single-episode instrumentation round, was 214 ms for observer routing ($n = 1034$ gated frames) vs. 437 ms per reactive step ($n = 345$); the $\sim 2\times$ advantage converts into success only when the prepared action is still valid at event time. No unsafe action occurred in either arm.

M Full limitations register

We state every carried caveat from the experimental record in full.

Claim scope. The positive result is bounded twice: it replicates within the Qwen3.5-MoE architecture class (a computer-use fine-tune and an untuned generalist), and everywhere else it is capability-gated. Nothing in this paper supports an architecture-agnostic claim, and three of the five non-Holo models we attempted show no gain at all.

Benchmark breadth. The main evidence chain (existence, confirmation, mechanism isolation, replication, ablations) runs on a single scenario family – the three-outcome `key_prompt` event – plus one external benchmark. The scenario was designed to isolate the mechanism, not to represent computer use broadly.

Knife-edge contested windows. Because reaction latency on the local benchmark is nearly deterministic, whether a 42-seed draw is contested at a fixed window is almost all-or-nothing: absolute success rates at a fixed window are not comparable across seed draws (the benchmark exhibits seed-set difficulty variance, not time drift). Only the paired per-seed comparisons carry evidential weight, and we caution against quoting this paper's absolute rates outside their seed sets.

Primary-window deviations. Two pre-registered primary windows missed their regime: the first contested scan's 700 ms primary landed at reactive ceiling and is not significant ($p = 0.25$) – the contested-regime result at that stage rests on pre-registered secondary windows and was subsequently confirmed by a fresh declared-primary run at 650 ms ($p = 1.8 \times 10^{-3}$) – and the ablation stage's 650 ms primary was recalibrated to 600 ms by the reactive-only protocol. Both deviations are reported where they occur.

A near-miss on a validity sub-target. The planner-hardening stage set a pooled-with-history Wilson lower bound of 97% on tree validity; the achieved bound is 96.69% (425/432 pooled). The pre-registered per-run acceptance (30/30 valid) passed, and we did not expand the sample to chase the pooled target.

Pre-compiled trees cannot bind late parameters. The

Table S6: The $k \times n$ frontier at 600 ms (shared R0 $= 0.548$). The node budget is the schema's maximum tree size; critical-path cost is the observer path in output-token equivalents (p50). Both k=3 cells dominate R0 (19 T2-only vs. 0 R0-only pairs); both are 42/42, but n=2 spends a $3.25\times$ node budget on an identical realized tree.

Cell	k	n	Budget	T2 succ. ↑	Validity ↑	T2-/R0-only	Crit. path (tok) ↓
$k1\,n1$	1	1	2	6/42 (0.143)	36/42 (0.857)	4 / 21	51.9
$k1\,n2$	1	2	3	1/42 (0.024)	5/42 (0.119)	1 / 23	54.1
$k2\,n1$	2	1	3	8/42 (0.190)	17/42 (0.405)	2 / 17	54.5
$k2\,n2$	2	2	7	6/42 (0.143)	12/42 (0.286)	5 / 22	54.7
k3 n1	3	1	4	**42/42 (1.000)**	**42/42 (1.000)**	**19 / 0**	58.8
$k3\,n2$	3	2	13	42/42 (1.000)	42/42 (1.000)	19 / 0	56.9

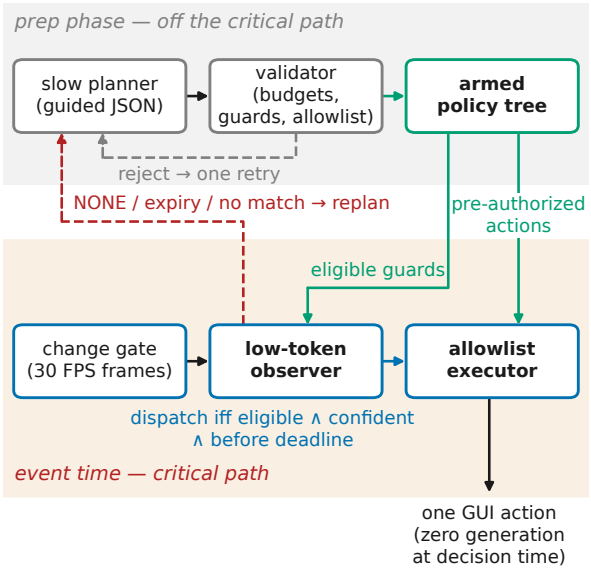

Figure S2: AAPT dataflow. During the prep phase (off the critical path) the slow planner compiles a policy tree under guided decoding; a validator enforces budgets, guards, and the action allowlist, with one bounded retry. At event time the critical path is a change gate, one low-token observer call, and a pre-authorized lookup – no generation. `NONE`, deadline expiry, or an unmatched frame falls back to replanning.

`click_target` scenario was not run as a paired comparison because both arms fail for independent reasons: the reactive arm is floored by the model's coordinate grounding (0/18 even at relaxed windows; clicks miss a 90 px target by a 229 px median despite the model perceiving it), and the AAPT arm is blocked by construction – the target's coordinates are revealed only at fire time, after the tree is compiled, and the observer returns a branch id, not coordinates. Positional grounding and observer-parameterized actions are unimplemented extensions.

External transfer is a conditional split over small samples. The DynaCU aggregate is a tie ($p = 1.0$, $n = 39$), and the three-episode power run (majority-of-3) confirms the category dissociation under a pre-committed concentration rule – but the split still rests on 4 vs. 5 discordant tasks, all AAPT-side wins sit in one category (E dashboards; the single-episode J wins did not survive the majority rule), and the result licenses a boundary claim, not a per-category effect size.

Table S7: Observer token-cap sweep, pooled over the two contested windows (eight cells, 42 pairs each).

Cap	Pooled T2	Rate	Note
32	48/84	0.571	truncates JSON (27 fail.)
64	**70/84**	**0.833**	winner: 0 fail., lowest p95
96	69/84	0.821	statistical tie with 64
128	58/84	0.690	ineligible: branch acc. 0.829

One boundary model has thin coverage. The UI-Venus evidence is two 42-pair blocks – the original tie (branch accuracy 0.39, outcome recall 0.22) and the oracle probe that reversed our committed prediction (Appendix G) – and the probe also exposed a between-session shift in the planner's tree coverage (recall 0.22 vs. 0.59 under identical config and compile key), so per-session tree statistics for this model should be treated as unstable. Similarly, UI-TARS's reactive 0/30 arises under the base reactive prompt after the single permitted adaptation round was spent on its planner; it is a harness-portability datum, not a capability ceiling for that model.

Observer headroom. The oracle gap ($\mathrm{O1} - \mathrm{T2} \approx 0.16$ at the contested windows) means a sizable fraction of compiled-tree value is lost in live routing; the reported T2 numbers understate what the trees themselves contain.

Unvalidated design components. Asynchronous tree refresh, reusable policy memory with promotion gates, and richer simulated environments are design sketches only (Appendix J); no experiment in this paper validates them. No fine-tuning was performed anywhere: the pre-registered training triggers (planner and observer LoRA gates) were never met, which we count as a strength of the frozen-model result but also means this paper says nothing about what training would add.

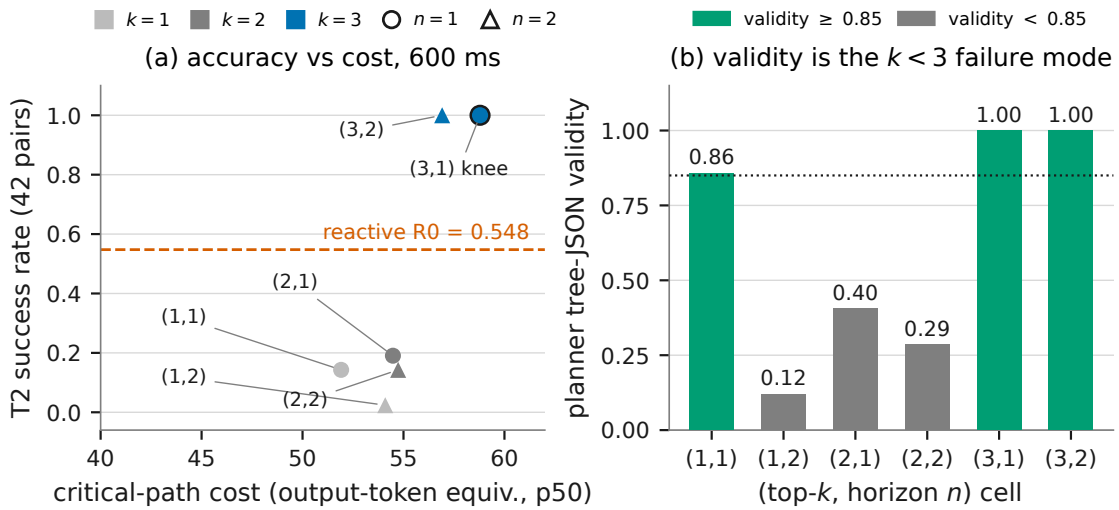

Figure S3: Stage E budget frontier at the 600 ms primary window (42 pairs per cell, shared reactive arm). (a) T2 success vs. critical-path cost (k = color, n = marker shape); the knee $(k{=}3, n{=}1)$ matches the branch budget to the three-outcome event set at minimal depth. The cost axis starts at 40 token-equivalents; the realized spread (52–59) is under 15%, so cost is nearly flat across cells. (b) Planner tree-JSON validity per cell against the 0.85 gate (dotted): budgets below the outcome-set size fail primarily by *validity*, with $k{=}2$ the worst cell.

Table S8: Continuation of the main paper's prediction ledger: entries added in the Round-1 revision cycle. As in the main table, every entry was declared before its block's data and failed predictions are retained at full weight.

Stage	Declared before data (primary endpoint)	Outcome
Five-outcome family	addendum: seeds, windows (T2>R0 @900 ms)	null (R0 ceilinged post-scan); exploratory 750 ms $p = 1.3 \times 10^{-3}$ (App. E)
Budget cells	$k \in \{1, 2, 3, 5, 7\}$, knee-at-5 (T2 vs. k)	confirmed: knee at $k{=}5$, valley at $k{=}3$ (App. E)
Uneven arm	drop-rarest ($\rho{=}0.95$, $K_{\max}{=}4$)	29/30 + 39/42 dropped exactly F9; paired arm nulled by τ_m prior collision (App. E)
Prep sweep	crossover at $L_{p95}{+}M \approx 2.4$ s (T2 vs. prep)	confirmed: 0/42 at 0.5–1 s, tie at 2 s, 42/42 at 4 s; all failures misses (App. F)
Delay jitter	T2 unchanged under $U(500, 1000)$ ms	confirmed: $p = 0.11$ n.s.; T2>R0 15/0, $p = 6.1 \times 10^{-5}$ (App. F)

Table S9: Contested-window scan (42 pairs per window). The pre-registered primary (700 ms) is n.s. because R0 is near ceiling there; the secondaries carry the contested-regime result. Incorrect actions: 0 everywhere.

Window	R0	T2	Disc. (T2/R0)	McNemar p
600 ms	4/42 (0.095)	29/42 (0.690)	25 / 0	6.0×10^{-8}
650 ms	12/42 (0.286)	30/42 (0.714)	19 / 1	4.0×10^{-5}
700 ms (prim.)	37/42 (0.881)	40/42 (0.952)	3 / 0	0.25 (n.s.)
750 ms	39/42 (0.929)	41/42 (0.976)	2 / 0	0.5 (n.s.)

Table S10: Per-trial cost accounting, declared-primary block (B.1, 650 ms, 42 pairs per condition). Completion tokens are means per trial; wall time is armed-to-terminal.

	R0	T2
Planner completion tok	0	283.5
Planner latency p50 (s)	–	1.81
Observer/decision completion tok	105.8	59.3
Total completion tok	105.8	342.8
Wall time per trial (s)	1.65	3.32
Trees per successful trigger	–	1.27
Successes	21/42	33/42

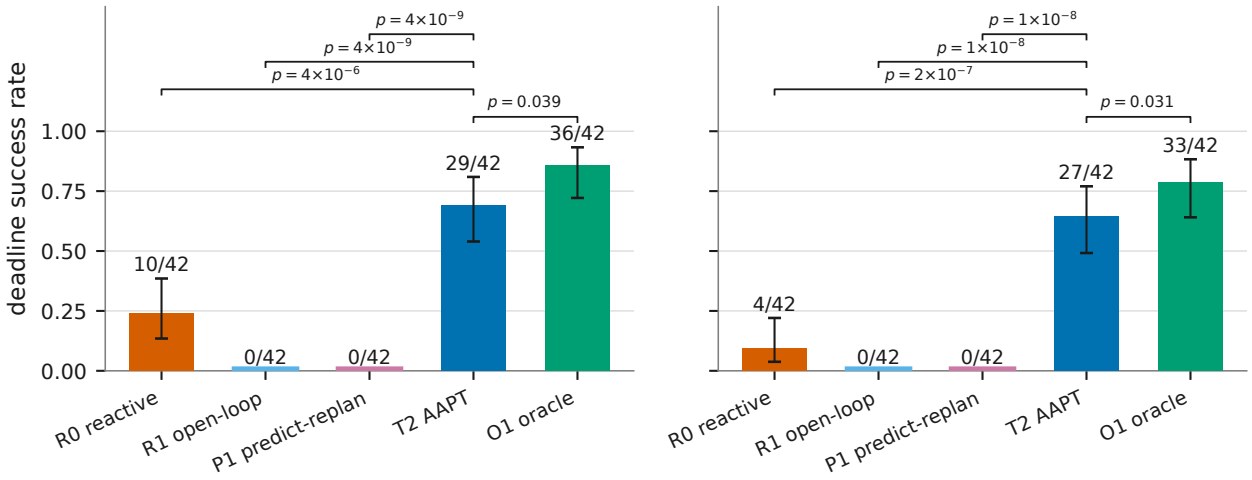

Figure S4: Mechanism isolation at the two contested windows (left: 650 ms; right: 600 ms; 42 pairs each; Wilson 95% intervals; brackets give exact McNemar p for T2 against each condition). R1 and P1 also act before a full round trip yet score zero – drawn as outlined stubs so 0/42 is a visible measurement – while O1 bounds what a perfect observer could extract from the same trees.

Table S11: DynaCU-Bench paired transfer, three-episode power run (majority-of-3 per task and condition; frozen Holo both arms; deterministic DOM scoring). Aggregate is a tie ($p = 1.0$); the discordant wins dissociate perfectly by whether the winning response was pre-enumerable when the tree was compiled (concentration rule pre-committed: both sides at 100% vs. an 80% bar).

Category	n	Reactive	AAPT	Discordant
D – carousels	10	3/10	0/10	3 react.-only
E – dashboards	9	0/9	4/9	4 AAPT-only
F – toasts/forms	10	4/10	2/10	2 react.-only
J – reaction games	10	0/10	0/10	0
Overall	39	7/39 (0.18)	6/39 (0.15)	5 / 4, $p = 1.0$

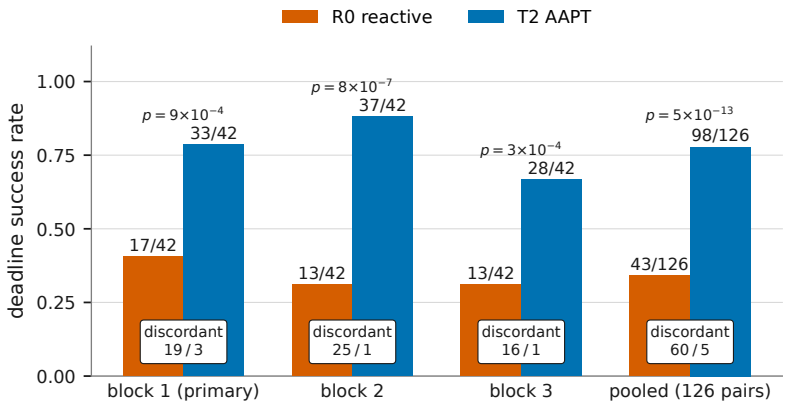

Figure S5: Pre-registered confirmation on Qwen3.6-35B-A3B, an untuned generalist of the same architecture class as Holo (42 pairs per block, 600 ms). Boxed annotations give discordant pairs (T2-only / R0-only), with the exact McNemar p above each pair. Every block is individually significant and the direction never reverses.