

Multi-Signal Coherence Verification: Preventing False-Positive Route Settlements in Autonomous Web Agents

Exzil Calanza

Independent research preprint. This work is not peer reviewed.

Source commit: `ecd71496f10ec95841123e956c5433e81dc310dd`

The reported counts characterize only the frozen experiments and production sample described here; they are not universal reliability estimates.

Abstract

Autonomous web agents often decide that a navigation or interaction has succeeded from one locally valid signal, such as the presence of a generic “ready” selector. That signal may establish that some page state exists without establishing that the browser reached the intended route or rendered the intended semantic state. This study evaluates a deliberately narrow route-coherence verification pattern in a preregistered synthetic benchmark. Six deterministic case families were generated with seed 158871: 100 correct controls and 500 drift cases comprising wrong-path, redirected-login, query-state, soft-404-at-the-expected-URL, and same-URL interstitial states. Every case preserved the same generic success selector. Three verifiers were compared: selector-only verification; selector plus a study-defined canonical final-URL check; and those two signals plus an application-specific semantic route anchor. Selector-only verification accepted all 500 drift cases. Adding the URL check reduced false positives to 200 of 500, while adding the semantic anchor rejected all 500 drift cases and preserved all 100 controls. Exact paired McNemar tests favored the multisignal verifier over selector-only verification (500 corrected discordant cases, 0 regressions, two-sided exact $p = 6.10987272699921 \times 10^{-151}$) and over URL-plus-selector verification (200 corrected discordant cases, 0 regressions, $p = 1.2446030555722283 \times 10^{-60}$). Because the benchmark is synthetic and the semantic anchor is deliberately informative, these results do not estimate real-world failure prevalence and do not establish universal sufficiency. They demonstrate a narrower engineering principle: verification signals should be selected to cover distinct failure surfaces rather than treating one tool-local success indication as proof of the end-to-end user outcome.

Index Terms— autonomous web agents, browser automation, route verification, false positives, semantic validation, URL coherence, WebDriver, Chrome DevTools Protocol.

I. Introduction

A browser automation step can succeed locally while the user-facing objective remains wrong. A selector may exist, a request may complete, a cache operation may return success, or a navigation API may settle, yet the final page can still be a login redirect, a stale route, a soft 404, an access interstitial, or another semantically incorrect state. This distinction is especially important for autonomous agents because downstream actions often rely on the previous step being not merely complete, but correct.

The motivating field note, “Three Green Signals, One Wrong Page,” documented a production-style verification problem in which evidence from deployed bytes, served bytes, cache behavior, and browser-resolved state answered different questions [1]. That operational observation motivates the present study but does not serve as evidence for its benchmark result. The experiment instead isolates one question: when intended and unintended browser states deliberately share a generic success selector, what additional protection is provided by final-URL coherence and an application-specific semantic route anchor?

The study compares three increasingly specific proof obligations. The first asks only whether a generic selector is present. The second also asks whether the observed URL is equivalent to the expected URL under a fixed study-defined canonicalization rule. The third additionally asks whether a stable semantic anchor matches the expected route identity. The benchmark was preregistered before execution, including its family counts, primary endpoint, acceptance criteria, and claim boundaries.

The contribution is intentionally limited. The study is not a measurement of production browser-agent failure rates, not a universal URL-equivalence proposal, and not evidence that three signals are sufficient for arbitrary web applications. It is a controlled demonstration that complementary signals can eliminate failure classes that remain invisible to a weaker verifier in the constructed cases.

II. Standards and Observation Surfaces

Modern browser automation exposes multiple observation surfaces rather than one canonical “page is correct” signal. The Chrome DevTools Protocol Network domain exposes request and response activity, timing, headers, bodies, and related network state. It also exposes separate controls for bypassing service workers and disabling browser cache, which demonstrates that browser-side cache behavior is not one undifferentiated state [2].

The WHATWG URL Standard defines parsing, serialization, origins, URL APIs, and interoperable processing requirements [3]. The benchmark in this paper does not implement or claim universal WHATWG URL equivalence. Its canonicalizer is an

explicit study rule used only to normalize the synthetic test cases.

W3C WebDriver similarly separates navigation and current-URL operations from element retrieval, page source, script execution, and screen capture [4]. That interface separation supports an engineering distinction central to this work: URL state, selector or DOM state, script-derived semantic state, network state, and rendered-pixel state are different evidence channels. None should be assumed to prove more than it actually observes.

III. Methods

A. Preregistered Design

The benchmark was preregistered before execution. It generated 600 deterministic cases using seed 158871:

- 100 correct controls;
- 100 wrong-path drift cases;
- 100 redirected-login drift cases;
- 100 query-state drift cases;
- 100 soft-404 cases that retained the expected URL; and
- 100 same-URL interstitial cases that retained the expected URL.

All 600 cases contained the same generic selector signal, [data-status='ready']. Ground-truth success was assigned by the generator before verifier evaluation and was not supplied to verifier logic.

The primary endpoint was false-positive rate among the 500 drift cases. Secondary endpoints were overall accuracy, false-negative rate among the 100 controls, per-family false-positive rate, and exact paired McNemar comparisons of the multisignal verifier against each weaker comparator.

The preregistered acceptance threshold required the multisignal false-positive rate to be lower than both comparators, zero multisignal false negatives among controls, both paired exact tests to satisfy $p < 0.05$, and exact agreement with the preregistered case and family counts.

B. Expected State

The synthetic expected route was:

`https://example.test/app/target?project=alpha`

The expected semantic anchor was a tuple encoding four application-specific fields:

`route:target | heading:Deployment Ready | action:Open Target | section:target`

This anchor is deliberately more informative than the generic selector. It is intended to model a stable semantic assertion that an application can define for an important route.

C. Verifiers

The three verifiers were:

V1 — Selector Only. Accept if the generic selector is present.

V2 — URL + Selector. Accept if V1 passes and the observed URL equals the expected URL after the benchmark-specific canonicalization function is applied.

V3 — Multisignal. Accept if V2 passes and the observed semantic anchor exactly matches the expected route anchor.

The study canonicalizer lowercases scheme and host, removes default HTTP/HTTPS ports, normalizes repeated path separators and dot segments, removes query parameters prefixed with `utm_`, sorts remaining query pairs, and removes fragments. These transformations are part of the experiment and are not presented as a general-purpose standard for URL equivalence.

D. Drift Families

The wrong-path, redirected-login, and query-state families altered URL state as well as semantic state. The soft-404 and interstitial families intentionally retained the expected canonical URL and generic selector while replacing the semantic route anchor. This split allowed the benchmark to distinguish failures detectable by URL coherence from failures that require stronger application-level evidence.

IV. Results

All preregistered case counts were reproduced exactly: 600 total cases with 100 cases in each of the six families.

A. Primary Outcome

Among the 500 drift cases:

Verifier	False Positives	False-Positive Rate
V1 Selector Only	500 / 500	1.00
V2 URL + Selector	200 / 500	0.40
V3 Multisignal	0 / 500	0.00

All three verifiers accepted all 100 correct controls, yielding a false-negative rate of 0.00 for each verifier.

Overall accuracy in this constructed benchmark was 100/600 (0.1667) for V1, 400/600 (0.6667) for V2, and 600/600 (1.0000) for V3.

B. Per-Family Behavior

The URL-plus-selector verifier rejected all wrong-path, redirected-login, and query-state drift cases. It nevertheless accepted all 100 soft-404 cases and all 100 same-URL interstitial cases because those families deliberately preserved the expected canonical URL and generic selector.

The semantic-anchor requirement in V3 rejected those 200 same-URL semantic failures. In the benchmark construction, the semantic anchor was changed whenever the route semantics were wrong, so V3's perfect result is structurally linked to the benchmark design. It should not be interpreted as evidence that a real-world anchor will always be stable or sufficient.

C. Paired Exact Comparisons

For selector-only verification versus the multisignal verifier, 500 cases were wrong under V1 and correct under V3, while no case moved in the opposite direction. The two-sided exact McNemar p-value was $6.10987272699921 \times 10^{-151}$.

For URL-plus-selector verification versus the multisignal verifier, 200 cases were wrong under V2 and correct under V3, again with no opposite-direction regressions. The two-sided exact McNemar p-value was $1.2446030555722283 \times 10^{-60}$.

The preregistered acceptance criteria therefore passed.

V. Discussion

The experiment demonstrates why “the selector exists” and “the intended route is verified” are not equivalent propositions. V1 was intentionally vulnerable because every drift case retained the generic selector. V2 added meaningful protection by rejecting failures that changed path or query state, but it could not detect states that were semantically wrong while retaining the expected URL. V3 added a route-specific semantic obligation and therefore covered the deliberately constructed same-URL failures.

The result supports a proof-obligation pattern rather than a fixed three-signal recipe. A verifier should first identify the claim it needs to prove, then select signals whose failure coverage matches that claim. For a route transition, useful signals may include final URL, application-specific content identity, authenticated session state, expected primary controls, network completion, rendered output, or accessibility state. The appropriate combination depends on the application and the consequence of a false positive.

The findings also illustrate a broader automation design rule: a successful tool operation should not silently expand into a stronger end-to-end claim. A network request completing proves something about transport. A selector match proves something about DOM state. A current URL proves something about navigation state under a specified normalization rule. A screenshot proves something about painted output at one moment. Robust automation keeps those meanings separate until

enough independent evidence supports the actual objective.

VI. Threats to Validity

First, the benchmark is synthetic. Its family distribution was designed to test verifier behavior, not sampled from production traffic. The observed false-positive rates therefore must not be presented as real-world prevalence estimates.

Second, ground truth is simple and exact. Production applications can contain legitimate redirects, localization, experiments, role-dependent content, transient loading states, and ambiguous route semantics.

Third, the semantic anchor is deliberately strong. It encodes route identity, heading, primary action, and section, and the benchmark generator changes that anchor for semantically incorrect states. Real semantic anchors may drift, disappear, vary by locale, or be unstable under A/B testing.

Fourth, the URL canonicalization function is benchmark-specific. It is not a claim of complete WHATWG equivalence and may be unsuitable for applications where query ordering, tracking parameters, fragments, or path normalization carry business meaning.

Fifth, the experiment includes no visual oracle. It does not test occlusion, clipping, layout defects, accessibility-tree state, or whether a human could actually use the rendered interface.

Sixth, it does not test network settlement timing. A route can be semantically correct while still having incomplete requests or delayed application state; those concerns require additional evidence.

These limitations constrain the conclusion to the constructed experiment.

VII. Operational Verification Pattern

A practical route-settlement verifier can implement the following sequence:

1. Define the intended postcondition before acting.
2. Check final URL or route identity under an application-appropriate rule.
3. Check one or more semantic anchors that identify the intended state rather than a generic “ready” marker.
4. Add network, authentication, accessibility, or rendered-pixel evidence when those dimensions are material to the user outcome.
5. Treat disagreement between signals as a failed or unresolved verification, not as permission to select the most convenient green indicator.
6. Preserve enough evidence to identify which obligation failed and route recovery accordingly.

This pattern is useful precisely because each signal has a bounded meaning. It encourages agents to verify the claim they need rather than infer end-to-end success from one component’s local result.

VIII. Conclusion

In a preregistered 600-case synthetic benchmark designed so that a generic success selector was present on both correct and incorrect states, selector-only verification accepted all 500 drift cases. Adding a study-defined canonical URL check reduced false positives to 200, and adding an application-specific semantic anchor rejected all 500 drift cases while preserving all 100 controls.

The strongest conclusion is narrow: complementary proof signals can cover failure surfaces that a weaker verifier cannot observe. The result does not show that route drift is common, that selector-only automation fails at a particular real-world rate, or that three signals guarantee correctness. Autonomous web agents should therefore treat verification as an explicit proof obligation whose evidence is matched to the user-facing claim.

IX. Reproducibility

The experiment was executed on ZEKE with the registered Python interpreter and seed 158871.

Preregistration SHA-256: 7a92bb22f2a381104106afeb27e9113de7f2d1547b0ee90e997d3d3d0234805

Benchmark implementation SHA-256: 2aa8c44f95c09bd96577644bc65787498898900f8eb914cbb6d4e3b7cd4d3afc

Registered summary SHA-256: 181051cfb09fe537f81ed5b892c85052fb44644a0c8800ecdc2ce34451ab38af

Registered runs CSV SHA-256: 75b22ab100eb71e3b42436f43ecb1e18fcb2892f733ed51fa906947d552c7325

Registered cases JSONL SHA-256: 4780573c0411aefcb6a573ba4095efdfb50941b0fcca2f39a245796f23b90290

The independent Gemini 3.6 Thinking research review approved progression to this draft with no required fixes. That review is scientific/editorial evidence only; it is not publication authorization.

References

- [1] E. Calanza, "Three Green Signals, One Wrong Page," exzilcalanza.info, Aug. 8, 2026. Available: <https://exzilcalanza.info/three-green-signals-one-wrong-page/>
- [2] Chrome DevTools Protocol, "Network domain." Available: <https://chromedevtools.github.io/devtools-protocol/tot/Network/>. Accessed: Oct. 4, 2026.
- [3] WHATWG, "URL Standard," Living Standard. Available: <https://url.spec.whatwg.org/>. Accessed: Oct. 4, 2026.
- [4] W3C, "WebDriver," Working Draft, Jul. 2, 2026. Available: <https://www.w3.org/TR/webdriver2/>. Accessed: Oct. 4, 2026.