

Research Article

A Suite That Does Not Discriminate: The Case for Result-Set Fields in Agent Conformance Records

Evgenii Arsentev¹

1. Independent researcher

Work in progress in the W3C Agent Conformance group has converged quickly, though without adopting anything, on a per-check design for conformance records: five outcome states, a closed cause vocabulary, a validator that rejects impossible state-cause pairs, and — as of 14 September 2026 — a separate qualifier recording whether a check that passed could have failed. That design handles the two failure modes the group set out to catch: a check that never ran, and a check that could not have failed.

This paper argues that a third failure mode survives it untouched, because the property in question belongs to no check. A result set can consist entirely of sound, correctly reported, individually falsifiable checks and still support no conclusion, because no check changed state across the conditions being compared. No per-check field can express that. One surveyed format has already reached for the level above and is the closest prior art to what I propose: it carries a report-level discriminates boolean, constrained in prose and unenforced by its schema. That no per-check field can carry a property of a set is true by construction. What a corpus adds is the adjacent case, and evidence that it occurs in published work unnoticed. In 36 runs of a coding agent across six context-clearing policies, the verification gate returned 4,086 tests and zero failures — a verdict with zero variance in all six conditions — while modelled cost over those same conditions differed by 33.5% between the most and least expensive of the six conditions — the spread between the largest and smallest of six sample means, and so an extreme order statistic rather than a property of any one policy. (The two *extreme policies*, clearing every task and never clearing, differ by 21.6%; the 33.5% figure pairs the most aggressive policy with the interior optimum.) Every individual green mark was, so far as the record shows, correctly reported. The set of them invited a conclusion the accompanying data contradicts.

Re-analysis of the released run-level data yields two further results, the second of which is negative and is reported as such. First, the denominator behind that constant verdict was itself not constant: per-condition totals range from 641 to 701 tests and per-run totals from 97 to 127, so the published description of the experiment as holding “the verification suite constant” is, on my own data, wrong. Second, the tempting reading — that longer sessions did less work and the gate failed to notice — is not supported: model calls fall far more steeply than tests do, generated output rises, tests per model call rise, the correlation does not survive removal of one endpoint condition, and it does not survive Bonferroni correction over the twelve columns examined. What the record

establishes is weaker and sufficient: the amount of checking varied by up to 30% from run to run, and the artifact reporting the result could neither express that variation nor resolve its cause.

The proposal is correspondingly narrow, and I first made it in a message to that group rather than here: a result set should carry whether any check changed state across the compared conditions, and the identity of those that did.

What this paper adds to that message is the corpus, the negative result, and a structural obstacle that appears to be new. Where the checks are authored by the run itself — as they were in my corpus — check identity does not survive across runs, and the comparison-level question is not false but *undefined*. A record that cannot say incomparable will say none, and a reader will hear equivalent.

Correspondence: papers@team.qeios.com — Qeios will forward to the authors

1. Introduction

A conformance report, a CI summary and an agent evaluation all end in the same place: a green mark, which is meant to license an inference. It licenses that inference only under conditions the report usually leaves unstated — that the check was capable of the other answer, that we know how many checks there were, and that the checks meant the same thing across everything being compared.

The W3C Agent Conformance group spent September 2026 making the first of those machine-readable, and got further than one might expect. Section 2 describes that work at length, because this paper depends on it and because it is easy to reinvent by accident: two of the three things I initially believed I was proposing were already on the list, posted by other people, before I wrote them down.

What remains after that subtraction is the comparison. Conformance records are per-unit artifacts and the machinery being built around them is per-unit machinery: a state for each check, a cause for each state, a qualifier for each verdict, a validator that checks the combination. But the question an operator actually asks is almost never about one check. It is *did this change break anything*, asked across two or more result sets. A suite in which every check is sound and every check passes in every condition answers that question with a confident and unjustified “no”.

Section 3 gives a worked corpus from published work of my own in which the question could not even be asked — the case Section 5.2 is about — with the cost data sitting in the same files showing the conditions were materially different. Section 4 sorts the failure modes against what the current design catches. Section 5 states the proposal and an obstacle to it. Section 6 is limitations, of which there are many, and Section 7 concludes.

A note on standing: I am a participant in the discussion I describe, several of the sources cited are my own work or my own posts to that list, and the empirical material is a re-analysis of my own published experiment in which I am the party embarrassed. Section 3.3 reports a result that runs against my own thesis, and Section 6 says what the paper does not establish.

A note on overlap with my own prior post, since a reader is entitled to ask what is new here. I raised the result-set level in a closing paragraph of a message to that list ^[1], together with the headline numbers used in Section 3 and the field shape proposed in Section 5.1 — “one boolean plus the identity of the checks that moved”. Those are reproduced here, not rederived. What is not in that post, and is the reason for this paper: the corpus analysis of Section 3, including two corrections to my own published report and the retraction of a causal claim I had made from it; the obstacle of Section 5.2, which is that the comparison-level question can be undefined rather than merely unanswered; and the resulting incomparable value. A reader who has read ^[1] and ^[2] has read the idea; they have not read the evidence, the failure, or the obstacle.

2. What the group has already built

This section summarises work by other people, in enough detail that a reader can see how little of what follows is mine.

The survey. Nicolás Rocchia opened with a survey of how existing formats represent a check that did not run — four questions over eight formats on 1 September ^[3], revised to five questions over ten on 4 September ^[4], with each participant’s row left blank for its author to fill and Anton Sokolov correcting his own rather than confirming it. An eleventh row was contributed later by Rouiada Abbas ^{[5][6]}. The surveyed formats are TAP 14, SLSA Verification Summary Attestation v1.0, the proposed in-toto predicate agent-decision/v0.1, IETF AR4SI, draft-sokolov-rats-aep-composition-06, agentops, disensor residue/v0.4, CTEF, the CCS conformance vectors, and Roel Schuurkes’s score-free evidence layer ^[7].

Two findings from it matter here. First, formats that all “have” a not-run state intend opposite things by it: AR4SI defines its principal non-verdict value as equivalent to silence — on Sokolov’s reading, relayed in the survey, AR4SI says “a Relying Party’s appraisal policy should not distinguish value 0 from no Trustworthiness Claim” ^[4] — while TAP instructs harnesses to list skipped points, and disensor requires that they be rendered. Rocchia’s framing of the axis is that it decides “who is allowed to look away”. Second, of the ten formats, **TAP alone separates the state of a check from the cause**, and it does so with free text.

Prior art on closed vocabularies. Schuurkes brought two pointers to the table that were not survey rows ^[8]: XCCDF 1.2 ^[9], whose Default Model drops four result values from the count entirely and treats error and unknown like a fail, so a rule never evaluated leaves the score; and CAP-1, draft-hillier-coverage-attestation-00 ^[10]. CAP-1 defines a closed list of eight dispositions for a unit left out of an examined count — `not_applicable`, `disabled_by_policy`, `unsupported_input`, `resource_exhausted`, `failed`, `unavailable`, `out_of_scope`, `withheld` — and refuses free text in their place on an explicit ground: “Free-text reasons **MUST NOT** be accepted in place of a disposition, because free text does not aggregate.” It further requires that a result containing a `failed`, `resource_exhausted` or `unavailable` unit must not be marked complete, and that a denominator declare its own basis, since “a denominator without a basis is an assertion rather than a measurement.”

CAP-1 also supplies a precedent this paper leans on. Its conformance class mutation-tests its own verifier — each of the eight rules is silenced in turn and the class must fail in each case — on the stated ground that “rules that are never exercised are decoration.”

Five states, and state separated from cause. Schuurkes answered the survey with a two-field design ^[8]: a state, “what a consumer may conclude, and the only field that decides what an aggregate counts”, and a cause, “what a reviewer needs to decide whether a gap is acceptable”. Rocchia’s consolidation ^[11] names the five states — pass and fail as verdict states; not-exercised, inconclusive and void as non-verdict states, respectively a check that never ran, one that ran and reached no conclusion, and one that ran and whose evidence cannot be trusted.

Two details matter later. Schuurkes argued that an aggregate should never be emitted without the complete denominator beside it, and that any escape value must “cost something a reader sees, for example its own count in the summary and a run that cannot be reported as complete”. Schuurkes also identified the gap that matters most for the cause vocabulary: nothing in CAP-1 fits void, because “every disposition describes a unit that was not examined or not disclosed, while void is a unit that was examined and whose evidence cannot be trusted” ^[8] — a point Rocchia’s consolidation attributes to him explicitly ^[11], and which Ives restated the same day ^[12]. Ives’s own addition was a further distinction inside that seam: the cause vocabulary must keep integrity failures separate from availability failures, since a signature that will not verify is a corruption signal while an unfetchable key set is an operations signal ^[12].

The foreclosed check. The sharpest empirical contribution to the thread is Rocchia’s mutation check over the 89 conformance vectors of his own corpus: relax a rule by dropping its errors from the validator’s output, re-run every vector, and compare verdict and fired-rule list against what was recorded ^[13]. For the twelve rules with at least one vector naming them, the vectors that flipped were exactly those vectors. For two rules, R3 and R8, no MUST-FAIL vector existed — and none could be written, because all three schema versions already forbid the exact condition each rule tests, so the validator returns schema errors and never reaches the rule layer ^[14]. The requirement is stated twice, in the schema and in the rule, and only the schema is ever observable; a verifier that never implements R3 or R8 passes all 89 vectors and returns identical output on every artifact that can exist. Ives’s formulation is the one to keep: that is worse than untested, because untested is at least a visible gap, and this is not visible at all ^[15].

Rocchia’s remedy is a demonstration rather than an assertion ^[16]: the exemption list is shared data, and each run rebuilds the artifact that would fire the rule and requires both halves — that the schema rejects it, and that the rule still fires with the schema layer bypassed. It runs in two implementations, Python and TypeScript, from one shared declaration with no shared proof code, and he reports that his own first version had exactly the bug the exercise is about, passing “on noise” against a reused artifact that two schema versions rejected on an unrelated field.

Falsifiability, and the contradiction it created. Ives put the principle plainly: a pass is only evidence if you also report whether the check could have failed ^[15]. Rocchia then identified that this cannot coexist with the earlier design ^[11]: under the two-field rule, verdict states carry evidence references *instead of* a cause, so a pass has no cell in which to

record its own saturation — “Evgenii’s case is unreportable in the schema we just agreed on.” His resolution keeps two fields but requires the cause on all five states with two disjoint value sets, the verdict set answering one question — could this check have produced the other verdict on some artifact the emitter was allowed to produce? — with the values `counterexample-in-suite`, `counterexample-absent`, `counterexample-impossible` and `not-established`. The referent is the artifact rather than the run, a choice he flags as arguable and picks because it makes foreclosure detectable. He pairs it with a table of illegal state–cause combinations rejected by the validator, and warns of his own design that “if an emitter can satisfy the pair table by always writing `not-established`, the table has become decoration.”

Ives had taken the opposite view the day before: the two fields, validated against each other, “covers your saturation case (checked, could-fail, passed) and the foreclosed case (not-run, precondition-unsatisfiable) in one schema” ^[15]. I argued the following day for a third field instead of a split cause domain ^[11], on the ground that folding a qualifier of `pass` into a vocabulary built for the absence of `pass` makes aggregation incoherent — a consumer counting causes would be counting passes and non-passes in one column. Ives’s reading is defensible if cause is allowed two disjoint domains, which is precisely what Rocchia proposed; the disagreement is about whether one field with two domains is one field. Schuurkes’s reply endorsed the separation without settling the names: “execution outcome and evidence of discrimination are different axes. I would keep that separation” ^[21]. He added two cautions I accept: evidence establishing foreclosure and evidence demonstrating discrimination establish different things, and an unproved-style value must allow that constructibility itself is unknown rather than implicitly asserting that a failing artifact exists.

Where that leaves the per-check record. As of 14 September the thread holds three axes — state, cause, and a discrimination qualifier — with field names unfrozen. Both failure modes it set out to catch are covered. A foreclosed check is `not-exercised` with `precondition-unsatisfiable`, and the pair table makes the coverage-count lie illegal. A saturated check is a `pass` whose qualifier says no counterexample is present, and the emitter is forced to answer. Rocchia’s worked rows include my own suite, correctly marked as awaiting an answer I had not yet given ^[11].

I record all this at length because the denominator rule I had thought was mine is Schuurkes’s ^[8], and the falsifiability vocabulary I proposed is one of two competing designs for an axis the group had already opened. This paper proceeds from that work rather than competing with it.

3. A corpus in which every check passed

The remaining gap is best shown on real data, so this section re-reads a published experiment of mine as a conformance artifact. Its design, cost result and mechanism are reported in full elsewhere ^[17] and are treated here as a premise, not restated as a result. The headline pairing below — a constant verdict beside a cost difference — I first stated in the message to the list cited here as ^[1], at the 25.3% given by the incorrect prices since corrected in ^[17]; it is reproduced rather than newly derived. What is new is the rest: three analyses of the released material that neither the

original report nor that message contains. Two are unflattering to the report; the third undercuts this paper's own thesis.

The setting. A fixed set of twelve programming tasks was executed by a coding agent under six context-clearing policies — a fresh session every 1, 2, 3, 4, 6 and 12 tasks — with six replicates each, holding the tasks and their order constant; 36 runs. Every run ended with the project's full test suite, treated as a gate rather than a secondary metric, on the stated reasoning that a cost comparison is worthless if the cheaper policy also did less work. Run-level counters for all 36 runs are open data [\[18\]](#).

What the gate reported. Across all 36 runs: 4,086 tests executed, 4,086 passed, zero failures. No run and no condition produced a red mark. The verdict has zero variance and can, by construction, support no statement about differences between conditions.

What sat beside it in the same files. Modelled cost per run moved from \$2.165 under the most aggressive clearing policy to \$1.622 at the observed optimum and back to \$1.780 when the context was never cleared (Table 1 of [\[17\]](#), rows *tasks per session* = 1, 3, 12). Exact two-sided permutation tests over all 924 splits give $p = 0.0022$ against the aggressive endpoint (0.011 after Holm adjustment over the five comparisons) and $p = 0.046$ against the never-clear endpoint (0.14 after adjustment, so that difference is not significant). I recomputed all five of [\[17\]](#)'s permutation tests from the corrected per-run costs of version 1.2 and recovered its reported values exactly.

Two cautions belong with those figures. The reference condition was selected as the observed minimum of six sample means, so the p -values are post-selection; version 1.2 of [\[17\]](#) states this and adds a Holm adjustment over the five comparisons, which does not account for how the reference was selected. And the 33.5% headline is the spread between the largest and smallest of six means, an extreme order statistic rather than a property of any policy. I quote them as the best available description of what a reader of thirty-six green reports was not told, not as a result this paper defends.

A consumer reading those thirty-six green reports concludes the six configurations are equivalent. The data in the same deposit says they are not. The gate was not wrong; it was uninformative, and nothing in its output said so.

How little a clean sweep constrains. By the rule of three with the run as the unit, zero failures in 36 runs is consistent with a per-run failure probability as high as 8.3% at 95% confidence. With the test as the unit, zero in 4,086 bounds the per-test probability at 0.073%. Both are correct and they differ by two orders of magnitude, which is the point: a pass count is uninterpretable until the record says what the unit was.

3.1. The denominator was not constant, and the published description is wrong

The 4,086 figure is a grand total over all 36 runs, not a per-condition constant. Per-condition totals are 701, 697, 687, 699, 661 and 641 tests for policies of 1, 2, 3, 4, 6 and 12 tasks per session; per run the count ranges from 97 to 127, a spread of 30% between the least- and most-checked run in the study.

Two sentences in the abstract of ^[17] do not survive this. It says “4086 tests, zero failures, in every condition”: the zero-failures half holds everywhere, the 4,086 half is a sum. (Its results section is more careful, saying only that across all 36 runs 4086 tests executed and 4086 passed.) More seriously, the same abstract states that the design held “the model, the tasks, their order, and the verification suite constant”. The verification suite was not constant. It was written afresh by the agent in every run and differed in size between runs by up to thirty per cent. I wrote that sentence, and I correct it here rather than let it stand.

3.2. *Why the suite is not an instrument*

All twelve task prompts instruct the agent to create the test file itself and to run `node --test`; nine of the twelve specify a minimum number of cases — between five and ten — and none specifies a maximum; three state explicitly that the suite must pass. The final gate therefore re-runs a suite the agent authored and had, in the course of completing each task, already satisfied. Nothing in the harness aborts, retries or discards a run on failure; the exit status is not consulted.

A floor with no ceiling — “at least six” in the first task, and a comparable minimum in eight others — is the design decision that turns the denominator into a free variable: an agent writing the minimum and an agent writing twice the minimum both comply, and the gate scores both as a pass. So the agent writes the criterion and satisfies it; the ordinary failure mode is spent inside each task before the gate runs; and a run that did less would not go red — it would write fewer tests.

This argument is structural and needs no statistics. A reader of the verdict cannot distinguish a run that passed 127 checks from one that passed 97, and cannot learn that the checks were chosen by the subject of the experiment. Under Schuurkes’s denominator rule ^[8] the first is already a defect; Section 5.2 argues the second is deeper.

The shape is not unique to my corpus. Rocchia found the same thing in another participant’s checker and verified it by running it: a check whose companion file is missing leaves no record behind, and the summary counts only checks that ran, so the same artifact reports 40/40 with the file and 38/38 without, passing both times ^[19]. His reading — the report gets smaller, not worse — is exactly the failure mode here, arrived at independently.

3.3. *The causal claim the data does not support*

The denominator does move with the treatment: mean tests per run falls from 116.8 under the shortest policy to 106.8 under the longest, a decline of 8.6%, with a Spearman rank correlation of -0.434 (two-sided permutation $p = 0.0092$, 200,000 shuffles, seed 20260914). The decline is not monotone — condition 4 sits above condition 3 — and I describe it as a decline on average with an interior bump.

It is tempting to read this as the cheaper policy doing less work unseen: precisely the failure the gate was installed to catch. An earlier draft of this paper made that claim. **The released data does not support it**, and three columns of the same file say so.

quantity	rho vs tasks-per-session	permutation p
tests executed	-0.434	0.0092
model calls	-0.812	< 0.00001
output tokens generated	+0.626	0.00006
tests per model call	+0.682	0.00001

Model calls fall far more steeply than tests, and that fall is close to structural: sessions per run is exactly twelve divided by the policy, and each fresh session must re-orient itself. Against that backdrop the test count falls *less* than the call count, so tests per model call **rise** with session length; generated output rises too. On both other volume-of-work proxies in the dataset, long-session conditions did more, not less.

Three further limits:

The correlation rests on one endpoint. Excluding the never-clear condition — six runs — takes rho to -0.281 at p = 0.132.

excluded	n	rho	p
none	36	-0.434	0.0092
every 1 task	30	-0.409	0.026
every 2 tasks	30	-0.475	0.0087
every 3 tasks	30	-0.495	0.0059
every 4 tasks	30	-0.495	0.0060
every 6 tasks	30	-0.373	0.043
every 12 tasks	30	-0.281	0.132

Multiplicity. The analysis is post-hoc; I examined rank correlations between the treatment and the numeric columns of the released dataset. The basis of that family should be stated rather than assumed, since a denominator without a basis is an assertion: the file has fourteen numeric columns, of which tests_failed is constant and admits no correlation, tests_passed duplicates tests_total, sessions is a deterministic function of the treatment and replicate is orthogonal to it by design. Counting the twelve columns on which a correlation is computable gives a Bonferroni threshold of 0.0042 at $\alpha = 0.05$; counting all thirteen besides the treatment gives 0.0038. **p = 0.0092 survives neither.** I

report the uncorrected value and this sentence together, because the uncorrected value on its own would misdescribe what the test establishes.

An untestable rival. The benign explanation — longer runs finished fewer tasks — is not testable from the deposit, which has no tasks-completed column. I excluded it only by inspecting the run directories on the machine that produced them: all 36 runs contain all twelve source modules and all twelve test files, so none is truncated. That check rests on material outside the public record, which is itself an instance of the problem.

I therefore withdraw the causal reading entirely, and with it any claim that the variation is systematic: on this evidence I cannot distinguish a real association from an artefact of a structurally imposed decline in model calls. What survives is a finding about the record, and it needs no p-value: a quantity that varied by up to 30% from run to run was invisible in the artifact reporting the result, and remains unresolvable from the data published beside it.

3.4. Which case is this

Rocchia asked directly whether the pass count could have differed or was constant by construction ^[14]. It could have differed: a regression late in a run can break tests written early, nothing re-verifies earlier tasks in between, so red is reachable. In his vocabulary my suite is pass with counterexample-absent — saturated, not foreclosed.

But my answer on the list ^[1] was more confident than the harness warrants, and the part I did not say is the part that matters. The gate's power against *the specific possibility it was installed to exclude* is close to nil, because that possibility lands in a denominator the gate does not report. A check can be formally falsifiable and still be inert against the hypothesis it was placed there to test.

I should also be explicit that I did not *prove* falsifiability. I inferred it from the construction of the harness rather than demonstrating it by relaxing something and watching a vector flip — the experiment Rocchia ran on his corpus and I did not run on mine. Schuurkes's caution applies with force here: an attached input alone does not establish that the running checker distinguished it ^[2]. Under either proposed vocabulary my study sits at the unflattering value, and that is correct.

Consistency also requires applying the same scrutiny to my premise. If the conditions differ in model calls and generated output, they were not equal-work conditions, and the per-run cost comparison inherited from ^[17] carries that caveat too. ^[17] notes the general form — the agent chooses how many calls to spend, so calls are an outcome rather than a controlled variable — but it would be selective to audit my gate and exempt my cost figure.

3.5. What the published record does not contain

Two identifiers any reader would need were absent from both artifacts as first published ^{[17][18]}; I supply them from the run logs. The model was invoked through a CLI alias resolving in the session logs to claude-sonnet-5, with no dated snapshot suffix, and it is the only model string present across the experiment logs. Modelled cost in versions 1.0 and 1.1 of ^[17] used prices of \$3.00 and \$15.00 per million input and output tokens, cache writes at \$3.75 and cache reads

at \$0.30; these were wrong. Version 1.2 uses the published list prices of \$2.00 and \$10.00, cache reads at \$0.20, and cache writes at the 1-hour rate of \$4.00 that the logs record for every write — the 20:1 ratio the mechanism in [17] turns on. The toolchain was Node v24.20.0 with Claude Code 2.1.252 on a Linux host.

Temperature was never set and is not recorded anywhere. Whatever the CLI default was is what ran; it cannot be recovered. I report no temperature rather than guess. Because the alias carries no dated suffix, the exact weights are likewise not pinned by the record.

None of this was concealed; no field required it, so it was never written down. That is the disease in miniature: an identifier no schema demands is one that silently fails to exist. I intend to deposit a corrected version of [17] carrying these identifiers and the corrections in Sections 3.1 and 3.3; until then the public record lacks them.

4. Three failures, and which the proposed design would address

Nothing in the table below is adopted; the per-check column describes what proposals currently on the list would do if the group adopted them, and field names are explicitly unfrozen [2].

failure	what it is	addressed per-check	how
foreclosed	no artifact that would fail the check can exist	yes	not-exercised + precondition-unsatisfiable; the pair table would make the coverage lie illegal [11]
saturated	the check could have failed and did not	formally, but see below	a pass whose qualifier records that no counterexample is present [11][1][2]
non-discriminating	every check sound and falsifiable, yet none changed state across the compared conditions	no	—

Row two deserves the qualification. The qualifier would label my suite correctly — pass, no counterexample present — and that label is true and nearly useless, for the reason Section 3.4 gives: the possibility the gate existed to exclude lands in a denominator the record does not carry. A per-check axis can be formally right about a check and silent about whether the check was worth running.

The third row is this paper’s subject. It is not a defect of any check and cannot be repaired at the level of a check.

And my corpus is not an instance of it. This is worth stating plainly, because the corpus is what motivated the paper and it is easy to over-read. Row three presupposes that a check in one condition is the same check in another; in my runs the agent wrote a fresh suite each time, so “no check changed state across the conditions” has no truth value

there. What my corpus instantiates is the obstacle of Section 5.2 — the case where the comparison-level question is undefined rather than answered “no”. Row three is established by construction, not by my data: a per-check field cannot carry a property of a set, whatever the set contains. The corpus shows something narrower and, for a proposal, more useful: that the undefined case occurs in practice, in published work, without anybody noticing.

One surveyed format does carry a discrimination signal, and the way it does so is instructive. The VRP conformance report ^[5] has two verdict states rather than one: alongside pass it has reject-as-required, recording that the engine caught something it was supposed to catch — evidence that the check can fail, of the observed rather than the claimed kind. More to the present point, VRP is the one surveyed format that already reaches the level above the check: alongside its optional aggregate it carries a discriminates boolean, and its prose says that flag is true only when every declared must-fail control was observed as reject-as-required, and that otherwise no aggregate may be published ^[6]. That is a report-level discrimination claim coupled to a publication bar — materially the shape of Sections 5.1 and 5.3, arrived at independently and earlier: the schema was merged on 2 September, before the exchange that produced this paper. Its weakness is not that the idea is unconstrained but that the constraint lives only in prose: the schema neither requires the field nor checks it against the declared controls, and a report omitting both aggregate and discriminates still validates.

Rocchia’s reading of VRP’s own published run is the per-check argument in miniature: three of five results are reject-as-required and therefore demonstrably can fail, two are pass, and “nothing in the record says whether those two could have failed” ^[6]. A verdict vocabulary carries the question for checks that caught something and is silent for checks that did not.

What a reader of my thirty-six reports needs is not in any check, however well qualified. It is a statement about the set: either that nothing moved, or — for this corpus — that the question cannot be asked of it.

5. The proposal, and an obstacle

5.1. *One field and one list, at the result-set level*

The proposal in this subsection is not new to this paper. I stated it on the list as “one boolean plus the identity of the checks that moved” ^[1], and what follows develops that sentence rather than arriving at it. The new material is Section 5.2.

A result set — the unit an operator actually compares — should carry whether any check changed state across the conditions compared, and the identity of the checks that did:

```

{
  "compared": ["ucurve-p1", "ucurve-p2", "ucurve-p3",
    "ucurve-p4", "ucurve-p6", "ucurve-p12"],
  "check_set_origin": "generated_by_run",
  "discrimination": "incomparable",
  "shared_checks": 0
}

```

That is the record my own corpus should have emitted, and the value is deliberately not the one a reader might expect. A suite in which nothing went red across six conditions looks like a case of none; Section 5.2 explains why none would be a false statement here and incomparable the true one. I put the awkward value in the worked example rather than the tidy one, because a proposal whose own illustration takes the flattering reading is not worth making.

discrimination takes one of three values — some, none, incomparable — and shared_checks reports how many checks the compared conditions actually have in common, beside the total. Four further things have to be pinned down before this is implementable, and I state them rather than leave them to a schema.

Zeroth, identity. For a fixed check set, two checks in different conditions are the same check when they carry the same stable identifier in the check set, not when their names happen to match; a format adopting this needs to say where that identifier comes from, and I do not settle it here.

First, the unit. A result set is the set of checks evaluated under one condition; compared names conditions, not runs. Where a condition has replicates, as all six of mine do, a check has a single state for a condition only if every replicate of that condition agrees; any within-condition disagreement disqualifies the check from the comparison and is reported separately. That is a deliberately blunt rule and a stricter one may be better, but a vaguer one is not implementable: without it, replicate flakiness reads as discrimination. Where no check survives the rule, the value is incomparable, not some.

Second, what counts as moving. “Changed state” must mean a change between the two verdict states. A check that is not-exercised in one condition and pass in another has changed state in the schema’s sense while establishing nothing about the conditions, so admitting it would let a flaky harness report itself as a discriminating suite.

Third, checks_that_moved under incomparable must be absent rather than empty, since an empty list is exactly what none emits and the two must not be confusable — which is the whole point of the value.

The list is the substantive member: a reader wants to know *which* checks discriminate, because those are the ones carrying the result. The summary takes a name rather than a boolean because it needs a third value a boolean cannot

express — and because VRP ^[5] shows what happens to a boolean whose meaning is carried in prose rather than in the schema: it can simply be omitted, and the report still validates.

The field's value is that it makes the null visible. A result set in which no check changed state is not evidence that the compared configurations are equivalent; it is evidence that this suite does not distinguish them, which is a statement about the suite. Schuurkes put the distinction more sharply than I had, in his reply to the message where I raised it: identical observed check outcomes mean no difference detected on those checks, “not equivalent configurations or absence of useful cost evidence” ^[2]. That is an endorsement rather than an independent arrival, and I cite it as one. Either value — none for a shared suite, incomparable for mine — would have blocked the inference the thirty-six green marks invite, while the cost figures in the same record differed by 33.5% on the extreme-order-statistic reading qualified in Section 3.

It composes with the per-check fields rather than duplicating them. none over a set whose checks are all foreclosed means nothing was ever going to move; none over a set of checks each carrying a demonstrated counterexample means a demonstrably capable suite looked and found nothing. Only with both levels present can a consumer tell those apart — and the second is the case where a null result is genuine evidence of equivalence rather than evidence of a blunt instrument.

5.2. The obstacle: check identity may not survive the comparison

The comparison-level question presupposes something the per-check design never has to assume — that a check in one run is a check in another. “Did any check change state across the compared conditions” has no referent unless the checks are shared.

In my corpus they are not. The agent authored its own suite in every run, and the runs contain between 97 and 127 tests; no stable identifier exists for a check across conditions, because no check exists across conditions. The honest value for my own study is therefore neither “something moved” nor “nothing moved” but **incomparable**, and a vocabulary of some / none / incomparable is the minimum that avoids recording a falsehood. The two failure states are not symmetric in consequence: a record that cannot say incomparable will say none, and none reads as equivalence.

This argues for a companion member on the check set giving its provenance. I name it `check_set_origin`, with three values — `fixed`, `generated_by_run`, or `mixed` — since that determines whether the comparison-level question is answerable at all. For `mixed`, the rule that follows from the above is that discrimination is computed over the shared subset and `checks_that_moved` may name only members of it, with `shared_checks` reporting the size of that subset; a `mixed` set whose shared subset is empty is `incomparable`. Three combinations are then detectable lies rather than judgements, in the manner of Rocchia's pair table ^[11]: `generated_by_run` with discrimination of some or none, since neither value has a referent when no check is shared; and `incomparable` carrying a `checks_that_moved` member at all, since an empty list there is indistinguishable from none and a non-empty one contradicts the state.

I should note an objection to my own framing here. Once discrimination can say incomparable, the provenance member supplies only the *reason* it is incomparable — and a reason attached to a state is what Section 2’s design calls a cause. The tidier proposal is therefore not a new enum at all but the group’s existing architecture lifted one level: a comparison-level state with a comparison-level cause, validated by a pair table. I have kept the explicit member here because it names the specific condition this paper is about, but a format adopting any of this should probably prefer the general shape. Nor does fixed by itself establish comparability: a fixed suite from which each condition selects differently is as incomparable as a generated one, so the member records how the set was obtained, not that the comparison is sound. The move is not new in kind: CAP-1 already requires a denominator to declare its *basis*, on the ground that “a denominator without a basis is an assertion rather than a measurement” ^[10]. Provenance is that requirement applied one level up, to a denominator assembled by the subject of the experiment. Schuurkes’s rule that an aggregate never appears without its complete denominator ^[8] is the same argument again.

The consequence for practice is worth stating plainly: where an agent authors the checks that judge it, cross-run comparison of verdicts is not merely noisy but ill-defined, and the record should say so rather than let a reader assume otherwise. This is a structural weakness of self-graded agent benchmarks independent of any statistics, and my own corpus is an instance.

5.3. *What the escape must cost*

The thread has converged on how to stop a vocabulary rotting — not as a group decision, but as a rule all three participants state independently — and it transfers directly. *incomparable* should carry the cost Schuurkes required of any escape value ^[8] and Rocchia attached to not-established ^[11]: its own count in any summary, and a comparison carrying one cannot be reported as complete. CAP-1 sets the same rule for incompleteness ^[10], and Rocchia’s own warning is the test to apply — if emitters can satisfy the requirement by always writing the escape value, the field has become decoration, and the check is whether the count of escapes rises while the count of complete comparisons stays flat ^[11].

I would add the obvious falsification for the proposal itself, in CAP-1’s spirit that rules never exercised are decoration: a corpus in which discrimination is none for every published comparison would show the field is being emitted rather than computed.

6. Limitations

One corpus, one toolchain, one operator. The empirical material is a single experiment on a single task set run by one person ^[17]. The reporting gap it illustrates is, I would argue, general; the magnitudes are not, and nothing here is a population estimate.

The re-analysis establishes less than it first appears. Section 3.3 is explicit: the denominator correlation is post-hoc, does not survive Bonferroni correction over the twelve columns examined, does not survive removal of one endpoint

condition, and is contradicted as a “less work” story by two other columns. I retain it only as evidence that the quantity varied at all, and that the record cannot resolve why.

My gate’s falsifiability is inferred, not demonstrated. The claim that my suite could have gone red rests on how the harness was built, not on a mutation experiment of the kind Rocchia ran ^{[13][16]} or CAP-1’s conformance class requires ^[10]. The repair is to run one.

No temperature, no dated model snapshot. Section 3.5 supplies what the logs retain and states plainly what they do not.

The proposal is untested and unimplemented. Nothing in Section 5 has been implemented by anyone, including me. The per-check design it builds on is an active mailing-list discussion rather than a group decision, with field names explicitly unfrozen ^[2]; both Internet-Drafts of mine cited here are individual submissions with no working-group adoption; and I am a participant in the discussion rather than a neutral observer.

An unsettled design question. Rocchia chose the artifact rather than the run as the referent for “could have failed”, flagging the choice as arguable ^[11]. The comparison-level field implicitly takes the *condition set* as its referent, and whether those two referents compose cleanly is a question I raise without settling. Schuurkes’s four test rows ^[2] are the right place to settle it, and they are not settled here.

7. Conclusion

The per-check half of this problem is close to solved, and was largely solved by other people while I was writing about it: five states, a closed cause vocabulary whose escape value costs something a reader sees, a validator that rejects impossible pairs, and a separate qualifier recording whether a pass could have been a failure. Together these make both a foreclosed check and a saturated one impossible to report as ordinary coverage.

What that machinery cannot express is a property of a result set rather than of a check, and that limit holds by construction rather than by measurement. My own corpus supplies the sharper half of the case: thirty-six runs, 4,086 tests, zero failures, every verdict true and every state-cause pair legal — beside a cost column showing the compared configurations differed by roughly a third of the bill on an extreme-order-statistic reading of six means, and a check count whose least- and most-checked runs differ by thirty per cent, for reasons the deposit cannot resolve. A consumer reading those verdicts concludes equivalence. Nothing in the record contradicts them — and in this corpus the record could not even truthfully say that nothing moved, because the agent wrote a different suite every run.

One field and one list at the result-set level would write it down. The obstacle is that where the checks are authored by the run itself, the question has no answer at all, and the vocabulary must be able to say so — because a record that cannot say incomparable will say none, and a reader will hear equivalent.

Notes

Ref. ^[9] was cited here as reported by Schuurkes in ^[8] and mapped by CAP-1's Appendix A ^[10]; I have not read the primary source, and say so rather than imply otherwise.

Version notes

This version corrects figures that this paper took from reference ^[17]. The modelled costs in versions 1.0 and 1.1 of ^[17] were computed at the wrong price level (\$3.00 and \$15.00 per million input and output tokens instead of the published \$2.00 and \$10.00) and with every cache write priced at the 5-minute rate, although the session logs record all writes as 1-hour writes. The error is the author's, and ^[17] has been reissued as version 1.2. As a result, the cost difference between the most and least expensive conditions changes from 25.3% to 33.5%, the difference between the two extreme policies from 8.4% to 21.6%, the condition means cited in Section 3 from \$2.680 / \$2.138 / \$2.473 to \$2.165 / \$1.622 / \$1.780, and the cache-write-to-read price ratio from 12.5:1 to 20:1; the difference between never clearing and the optimum, given here before as $p = 0.0108$, is no longer significant ($p = 0.046$ unadjusted, 0.14 after Holm adjustment). Test counts, model calls, token counters and every analysis in Section 3.3 are unchanged, and so is the thesis: the verification verdict had zero variance across all six conditions.

Statements and Declarations

Potential Competing Interests

The author is the author of the reports, drafts, software and dataset cited as ^[17], ^[20], ^[18], ^[21] and ^[22], and a participant in the W3C Agent Conformance group discussion cited as ^[19], ^[3]^[4]^[8]^[12]^[14]^[11]^[1] and ^[13]^[15]^[5]^[6]^[7]^[16]^[2]; six of the twenty-two references are his own work or his own posts to that list, and the paper's empirical material is a re-analysis of his own published experiment. Section 2 records that two proposals he initially believed original were anticipated on that list by others. Readers should also note the converse: the proposal in Section 5.1, and the pairing of a constant verdict with a cost difference used to motivate it (given there as 25.3%, at the incorrect prices since corrected in ^[17]), were stated publicly by the author in ^[1] one day before this paper was written, and are reproduced here rather than independently derived; reference ^[2] is a reply to that message, not an independent arrival at the same conclusion. The contribution this paper makes beyond ^[1] is the corpus analysis of Sections 3.1-3.5 and the obstacle and incomparable value of Section 5.2; Section 5.3 transfers to that value an escape-cost rule due to Schuurkes ^[8], Rocchia ^[11] and CAP-1 ^[10] rather than proposing one. No funding was received and there are no financial competing interests.

Data Availability

The run-level dataset re-analysed in Section 3 is openly published under CC BY 4.0 ^[18] and deposited with the report on Zenodo ^[17]. All statistics in Section 3 were computed from that file: permutation tests over conditions enumerate all 924 six-versus-six splits exactly; rank-correlation tests use 200,000 random shuffles under seed 20260914. The harness details in Sections 3.2 and 3.5, and the completeness check in Section 3.3, were read from the experiment's task definitions, run scripts, run directories and session logs, which are **not** part of the public deposit; a reader of the deposit alone cannot reproduce those three statements. All mailing-list sources are in the public W3C archive at the URLs given.

Author Contributions

CRedit: Evgenii Arsentev — Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing (original draft), Writing (review and editing).

Use of Generative AI

Generative AI (Claude, Anthropic) was used to draft and edit the language of this manuscript and to write the analysis code behind Section 3. No AI-generated figures appear. Every figure is computed by script from the openly released dataset ^[18] or quoted from a cited source with its location given, and the author is solely responsible for the accuracy, integrity and originality of the work.

Acknowledgements

Most of what this paper depends on was worked out by other people, and one part of it was not. Nicolás Rocchia, Kenne Ives and Roel Schuurkes designed the per-check record described in Section 2, and the first two rows of Section 4 are their result rather than mine; the comparison-level argument of Section 5.1 is mine, first stated on the same list ^[1] and developed here — though Abbas's format ^[5] had already coupled a report-level discrimination claim to a publication bar, which Section 4 credits. Ives's formulation that a pass is only evidence if you also report whether the check could have failed ^[15] is the sentence this paper is built around; Rocchia's question about whether my own pass count could have differed ^[14] is why Section 3.4 exists; Schuurkes's distinction between no difference detected and configurations being equivalent ^[2] is the argument of Section 5.1 stated better than I state it. The worked case in Section 3.2 is Rocchia's inspection of Guigui Wang's checker, reported in ^[19]; the one in Section 4 is Rouiada Abbas's VRP format ^[5]. Both are borrowed, and both belong to their authors.

References

1. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0028.html>, kArsentev E (2026). "W3C Public-Agent-Conformance, Message 0028, 14 September 2026." W3C public-agent-conformance. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0028.html>.

2. a. b. c. d. e. f. g. h. i. j. k. Schuurkes R (2026). "W3C Public-Agent-Conformance, Message 0029, 14 September 2026." W3C public-agent-conformance. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0029.html>.
3. a. b. Rocchia N (2026). "How Existing Formats Represent a Check That Did Not Run: A Survey, as Input for the Reporting Format." W3C public-agent-conformance, message 0000, 1 September 2026. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0000.html>.
4. a. b. c. Rocchia N (2026). "Survey, Second Version (Five Questions, Ten Rows)." W3C public-agent-conformance, message 0008, 4 September 2026. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0008.html>.
5. a. b. c. d. e. f. g. Abbas R (2026). "VRP Conformance-Report v0.1, Contributed as Survey Row Eleven." W3C public-agent-conformance, message 0026, 14 September 2026. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0026.html>.
6. a. b. c. Rocchia N (2026). "W3C Public-Agent-Conformance, Message 0027, 14 September 2026." W3C public-agent-conformance. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0027.html>.
7. a. b. Schuurkes R (2026). "Proposal: A Score-Free Conformance Evidence Layer." W3C public-agent-conformance, message 0007, 1 September 2026. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0007.html>.
8. a. b. c. d. e. f. g. h. i. j. Schuurkes R (2026). "W3C Public-Agent-Conformance, Message 0020, 10 September 2026." W3C public-agent-conformance. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0020.html>.
9. a. b. Waltermire D, Schmidt C, Scarfone K, Ziring N (2012). "Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2." NIST IR 7275 Rev. 4.
10. a. b. c. d. e. f. Hillier JD (2026). "The Coverage Attestation Profile (CAP-1)." Internet-Draft draft-hillier-coverage-attestation-00, IETF, 20 August 2026. <https://datatracker.ietf.org/doc/draft-hillier-coverage-attestation/>.
11. a. b. c. d. e. f. g. h. i. j. k. l. Rocchia N (2026). "Reporting Format, First Piece: State and Cause, and the Pair Table." W3C public-agent-conformance, message 0025, 13 September 2026. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0025.html>.
12. a. b. c. Ives K (2026). "W3C Public-Agent-Conformance, Message 0021, 10 September 2026." W3C public-agent-conformance. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0021.html>.
13. a. b. c. Rocchia N (2026). "Mutation Check Over 89 Conformance Vectors." W3C public-agent-conformance, message 0013, 8 September 2026. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0013.html>.
14. a. b. c. d. Rocchia N (2026). "W3C Public-Agent-Conformance, Message 0023, 13 September 2026." W3C public-agent-conformance. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0023.html>.
15. a. b. c. d. e. Ives K (2026). "W3C Public-Agent-Conformance, Messages 0017 (9 September 2026) and 0024 (13 September 2026)." W3C public-agent-conformance. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0017.html> · <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0024.html>.
16. a. b. c. Rocchia N (2026). "Proved Exemptions in Two Implementations." W3C public-agent-conformance, message 0018, 9 September 2026. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0018.html>.

17. ^{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s}Arsentev E (2026). "Clear Every Third Task: A Measured U-Curve in the Context Economy of Coding Agents." Technical report, ARSENTEV.AI. Version 1.2. doi:[10.5281/zenodo.22759217](https://doi.org/10.5281/zenodo.22759217).
18. ^{a, b, c, d, e}Arsentev E (2026). "Context U-Curve: 36 Coding-Agent Runs Under Six Context-Clearing Policies." Dataset, CC BY 4.0. doi:[10.57967/hf/10366](https://doi.org/10.57967/hf/10366).
19. ^{a, b, c}Rocchia N (2026). "W3C Public-Agent-Conformance, Message 0011, 7 September 2026." W3C public-agent-conformance. <https://lists.w3.org/Archives/Public/public-agent-conformance/2026Sep/0011.html>.
20. ^AArsentev E (2026). "The Context Economy of Agentic LLM Sessions: Where the Money Actually Goes." Technical report, ARSENTEV.AI. Version 1.1. doi:[10.5281/zenodo.22759216](https://doi.org/10.5281/zenodo.22759216).
21. ^AArsentev E (2026). "Agent Run Metrics: A JSON Interchange Format for Resource Accounting of Language-Model Agent Runs." Internet-Draft draft-arsentev-agent-run-metrics-00, IETF. <https://datatracker.ietf.org/doc/draft-arsentev-agent-run-metrics/>.
22. ^AArsentev E (2026). "Contextburn: A Run-Efficiency Meter for Coding Agents." Software. doi:[10.5281/zenodo.22712985](https://doi.org/10.5281/zenodo.22712985).

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.