Re-Perceived Effects: A Well-Formedness Contract for Accountable Actuation

Zain Dana Harper
Seattle, WA | ORCID 0009-0001-7175-5393
https://github.com/HarperZ9

July 2026

Abstract

When an agent actuates—writes a file, runs a command, moves an effector—the record that the action succeeded is usually the actuator's own report. This note describes a small seam that replaces the self-report with a re-perceived effect: the verdict that an actuation matched its intent comes from reading the world back, not from what the actuator said it did. The seam has two symmetric halves, each expressed as an independent judging criterion over a shared verification operation. A *gate* judges a proposed action against the capabilities actually granted, before execution. An *effect* criterion judges the world re-perceived after execution against the action's declared intent: for a file write, the on-disk hash against the intended hash; for a process, the return code; for a read, the witnessed bytes. Both criteria are total (a malformed input degrades to UNVERIFIABLE, never an exception) and neither trusts the actuator. We are explicit about altitude: this is a well-formedness contract for actuation—did the effect the world shows match the intent that was admitted?—not a result in robotics, control theory, or any physical domain. Its value is that it makes "the action did what it claimed" a re-checkable fact rather than an assurance, and it composes with a stability-claim validator that holds control-certificate assertions to the same must-fail discipline.

1 The self-report problem

An actuator reports success. A file-writer returns "ok"; a command runner returns a status it computed; an effector controller reports that it reached the setpoint. Each of these is the acting component vouching for its own action, and each can be wrong—or, under adversarial or buggy conditions, dishonest—without anything downstream noticing. The gap is the same one that appears wherever a producer certifies its own output: the certificate and the thing it certifies share an author. For actuation the fix is unusually concrete, because the world itself is available to re-read.

2 The seam

We express actuation accountability as two judging criteria over a single verification operation (*perceive, judge against an independent criterion, carry a re-checkable certificate*). The two criteria are deliberately symmetric.

Gate (before execution). Given the set of granted capabilities, a gate criterion judges a proposed action: is this channel and this target within what was granted? The action is admitted or refused *before* it runs, and the admission is itself a certificate.

Effect (after execution). Given the executed action's declared intent, an effect criterion judges the world *re-perceived*:

- **File write:** the intent carries the content that was meant to land; the criterion re-reads the file off disk, hashes it, and compares the on-disk hash to the intended hash. VERIFIED iff they match; REFUTED on any difference. The actuator's "I wrote it" is never consulted.
- **Process execution:** the re-perceived return code decides the verdict; a non-zero code is REFUTED, and a head of captured output is carried as evidence.
- **File read:** the witnessed bytes' hash is recorded as evidence of what was actually read.
- **Unknown channel:** UNVERIFIABLE with a stated reason—the seam refuses to pass a channel it cannot re-perceive, rather than assume success.

Both criteria are *total*: a malformed action or perceived state yields UNVERIFIABLE, never a raised exception, so a caller cannot crash the check into an implicit pass.

3 Why re-perception, not self-report

The design commitment is that the verdict is a function of the world read back, not of the actuator's claim. This has the same shape as re-derivable verification elsewhere—content addressing [2], proof-carrying results [1], reproducible checks [3]—applied to the effect of an action rather than to a stored artifact. Its practical force is that the actuator has no channel through which to launder a failed write into a success: the on-disk bytes either hash to the intended value or they do not, and the criterion that checks them did not author the write.

4 Composition with control-certificate claims

The same discipline extends to a stability-claim validator that checks assertions of the form "this trajectory satisfies this certified property." The load-bearing rule there is a *must-fail* fixture: every validated claim ships with a deliberately-broken counterpart the validator is required to reject, so a passing validation demonstrates the check is not vacuous on that shape. This is the actuation analogue of a verifier that must be shown able to fail; it holds a control-certificate claim to the same re-checkable, non-vacuous standard as the file-write effect.

5 Honest scope

We state the altitude plainly. This is a *contract*, not a physics result. The effect criterion witnesses that the bytes on disk match the declared intent, or that a process returned zero; it does *not* establish that the intent was correct, that a controller is stable in any control-theoretic sense, or that a physical maneuver was safe. Those are the actor's responsibility, and the seam makes no statement about them. What it contributes is narrow and real: a well-formedness contract that turns "the actuation did what it claimed" from an assurance the actuator issues into a fact a third party re-derives from the world, admitted before the fact by a capability gate and, for the control-claim case, held to a must-fail discipline. Prior verified-control and provenance work does not, to our knowledge, ship a sealed, offline-re-verifiable receipt binding a re-perceived effect to a declared intent at this altitude; that packaging, not a new control method, is the contribution.

References

[1] Necula, G. C. *Proof-carrying code.* POPL, 1997.

[2] Merkle, R. C. *A digital signature based on a conventional encryption function.* CRYPTO, 1987.

[3] Reproducible Builds Project. *Reproducible Builds: a set of software development practices.* https://reproducible-builds.org, 2026.