Why a ‘mechanical’ step failed the risk gate

Mullet’s risk gate fails a step labelled mechanical when its diff changes a literal or a call. A small prompt edit that has to change both failed it twice, so the label has to follow what the edit does, not how small it looks.



Published
1 November 2026

Reading time
2 minutes

About
Mullet

Tags
AI, Python, Testing

More on: Mullet

On 1 November a Mullet step labelled mechanical failed its risk gate twice on a small prompt edit. The gate scans the diff for changed literals and calls, and a step that has to change them is not mechanical, however small it looks.

In Mullet, every step in a job has a kind: mechanical, design or analysis, and a step with child steps counts as an integration step. Mechanical steps can go to cheaper workers, which is why the label is tempting, and the kind changes how some gates treat the step. The gates are described in Gates, not a model’s word.

What the gate does

Gate G3 fails a step of kind `mechanical` when its diff to an existing file does any of these:

- changes a literal
- changes the arguments of a call
- changes a comment
- adds control flow

- changes the value in an assertion
- flattens curly quotes to straight ones

It is a deterministic scan of the diff. No model reads it. The scan exists because cheaper models sometimes silently corrupt a literal or add a call argument. New files are skipped, because there is no existing code for the diff to change.

The message I saw was “4 risky edits in a mechanical node”.

What the job asked for

The job had three small prompt changes, each labelled mechanical:

- A brief pointer: add a sentence telling the worker where its brief is on disk, which means two new functions, an inline string replaced by a function call, and a changed function signature.
- A reviewer change: lower `MAX_FINDINGS` from 20 to 5 and add a paragraph to the reviewer’s instructions.
- A planner change: add guidance text.

Only the first step ran. It was blocked after four attempts: two aborted because the prompt did not fit the context window, and two failed gate G3. Both G3 failures were flagged for the same four edits, one call whose arguments changed and three string literals. The later gates never ran on either, so I do not know whether the code was otherwise right.

I think a correct version of that step would trip the scan as well, because the brief itself asks for a replaced string and a changed call. I have not tried a hand-written version. The other two steps never started, so I have no result for them.

For contrast, the same model passed a step that only created new files, with nine tests added. Nothing in that diff touched existing code, so there was nothing for G3 to flag. That is a sample of one.

Label by what the edit does

A mechanical step is one a cheap worker can do without judgement, and a changed literal or call is where silent mistakes hide. Size says nothing about that. A change like `MAX_FINDINGS` from 20 to 5 is one character and changes behaviour, and I would expect the scan to flag it, though that step never ran.

So the kind of a step has to follow what the edit does, not how small it looks. If a step changes values, strings or calls in existing code, it should not be labelled mechanical.

The code is at https://github.com/tomgould/mullet-public (https://github.com/tomgould/mullet-public).