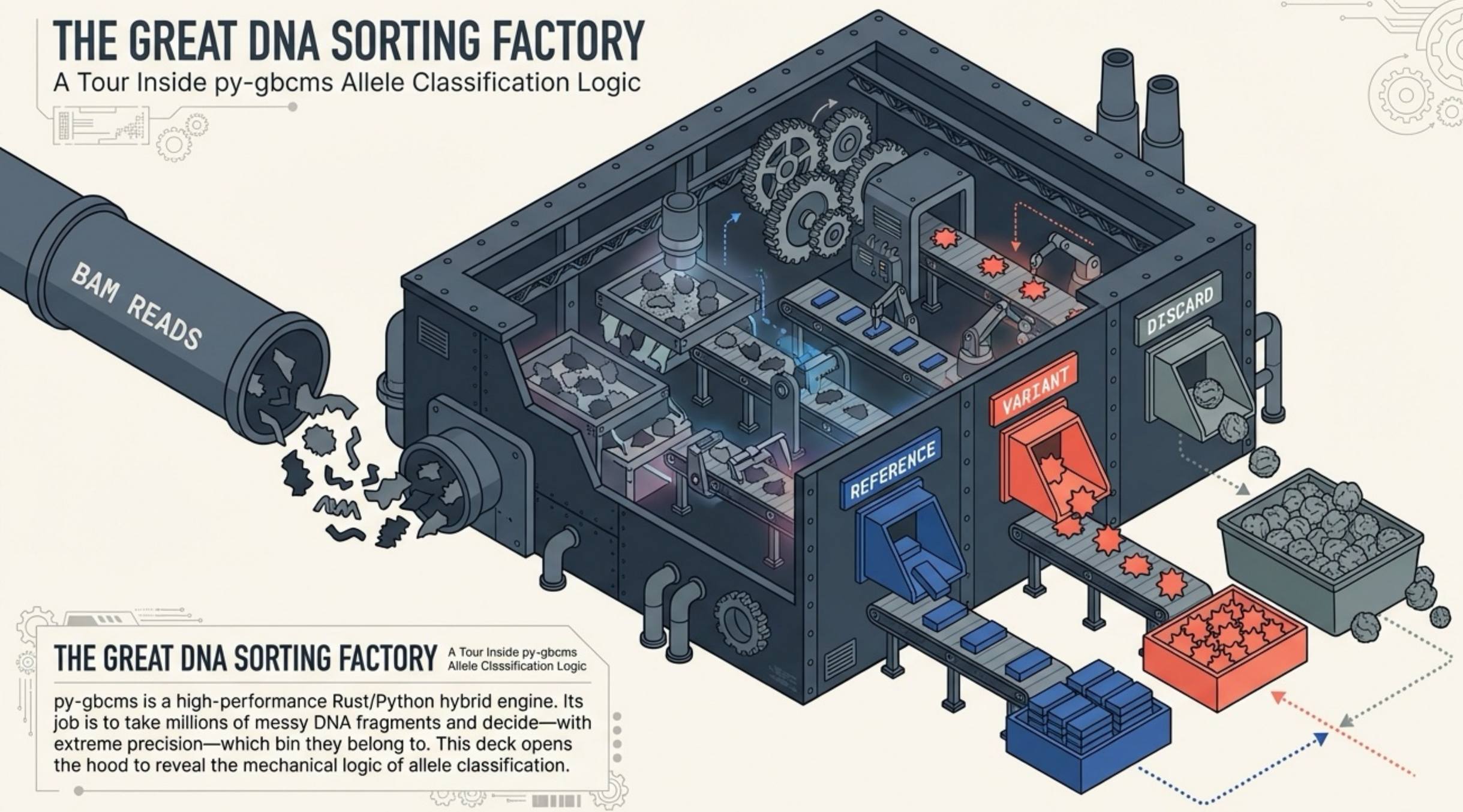
# THE GREAT DNA SORTING FACTORY

A Tour Inside py-gbcms Allele Classification Logic




THE GREAT DNA SORTING FACTORY A Tour Inside py-gbcms Allele Clsssification Logic

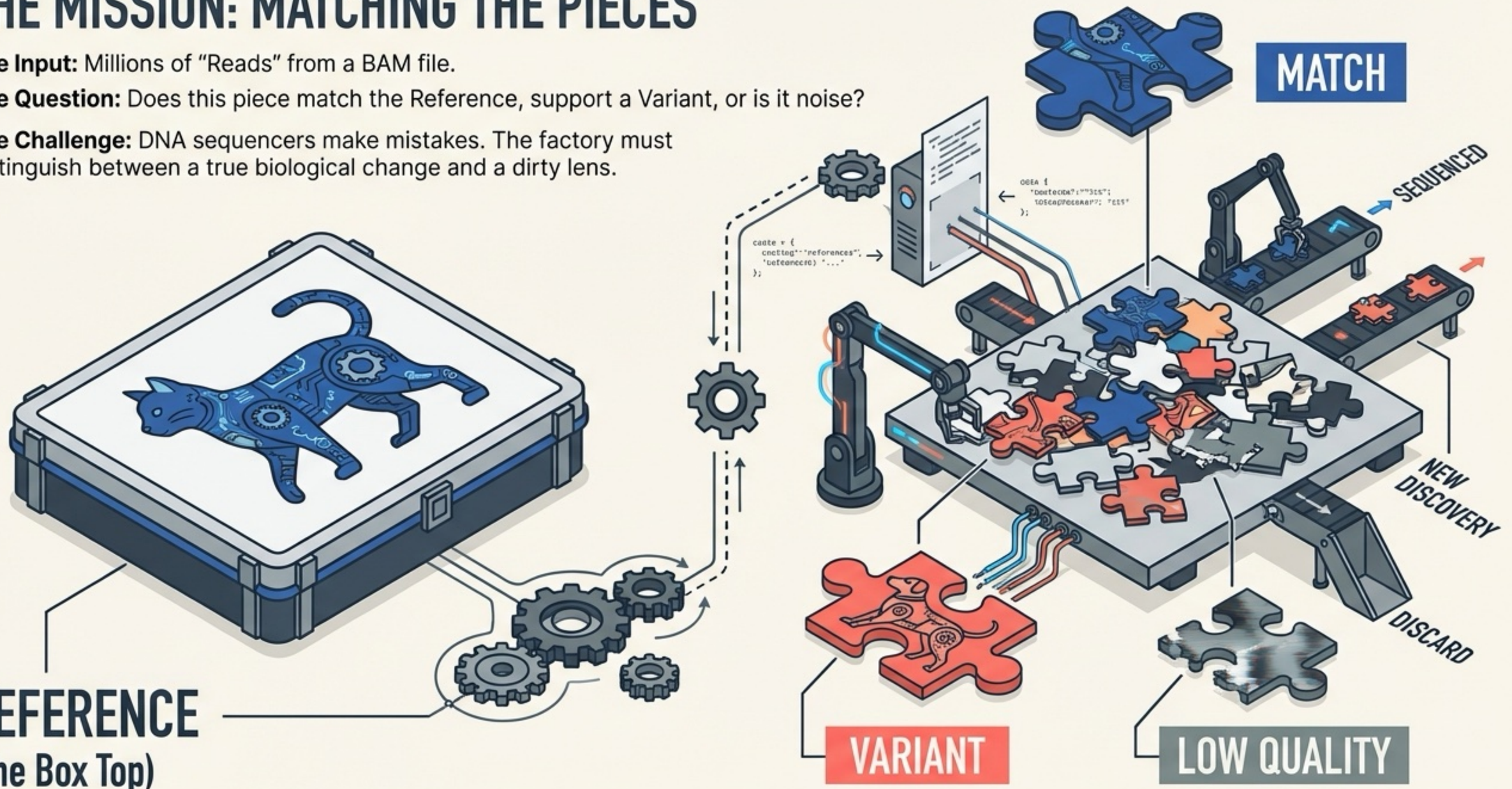
py-gbcms is a high-performance Rust/Python hybrid engine. Its job is to take millions of messy DNA fragments and decide—with extreme precision—which bin they belong to. This deck opens the hood to reveal the mechanical logic of allele classification.

# THE MISSION: MATCHING THE PIECES

**The Input:** Millions of “Reads” from a BAM file.

**The Question:** Does this piece match the Reference, support a Variant, or is it noise?

**The Challenge:** DNA sequencers make mistakes. The factory must distinguish between a true biological change and a dirty lens.

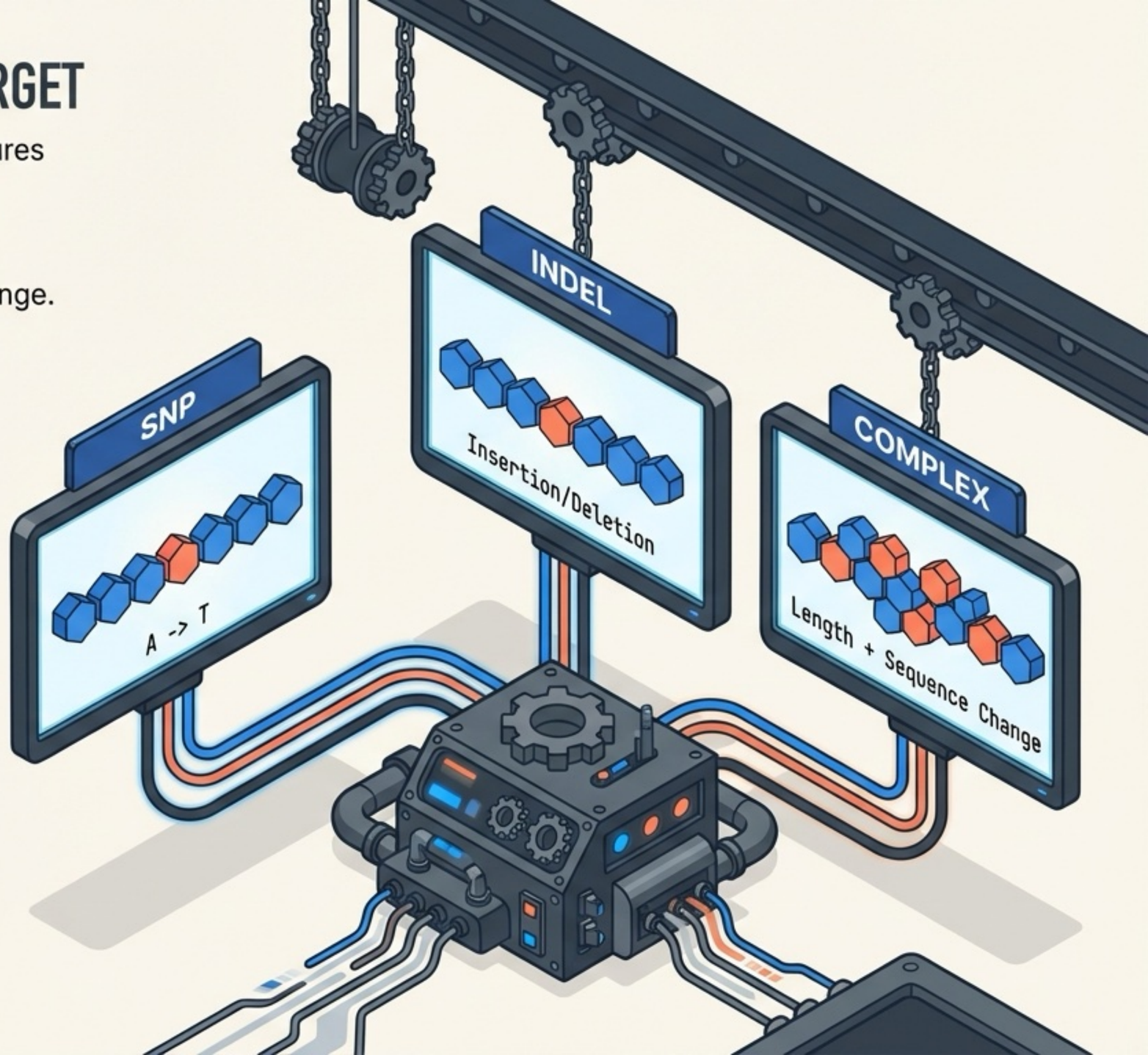


# THE BLUEPRINT: DEFINING THE TARGET

Before a read enters the system, py-gbcms configures its tools based on the variant type string.

* SNP: Simple substitution.
* Indel: Insertion or Deletion of bases.
* Complex: Simultaneous length and sequence change.

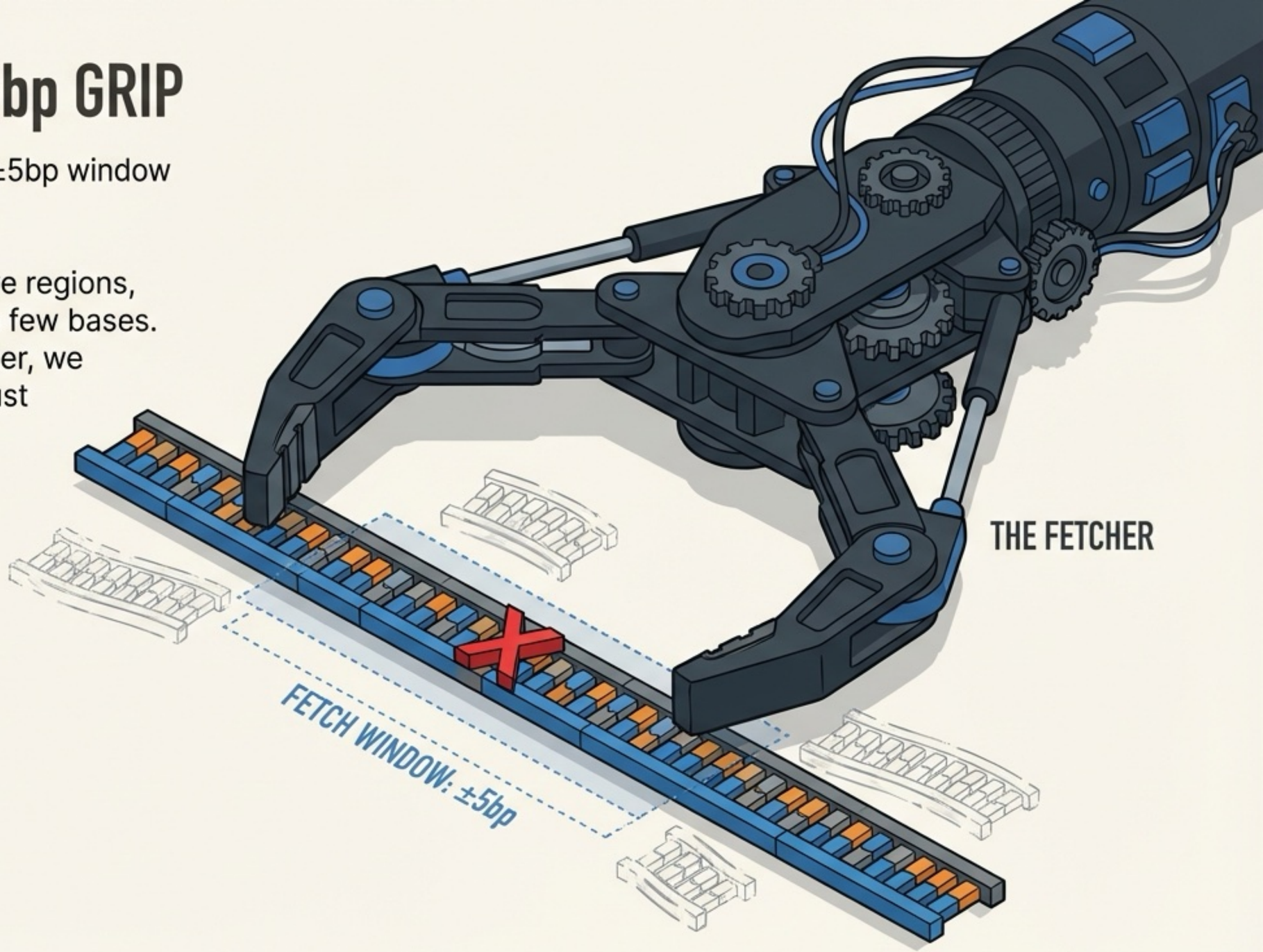
*Factory Note: If the type is unknown, the machine auto-detects MNPs or defaults to the Master Solver.



# THE INTAKE: THE ±5bp GRIP

The Logic: We fetch reads from a ±5bp window around the anchor.

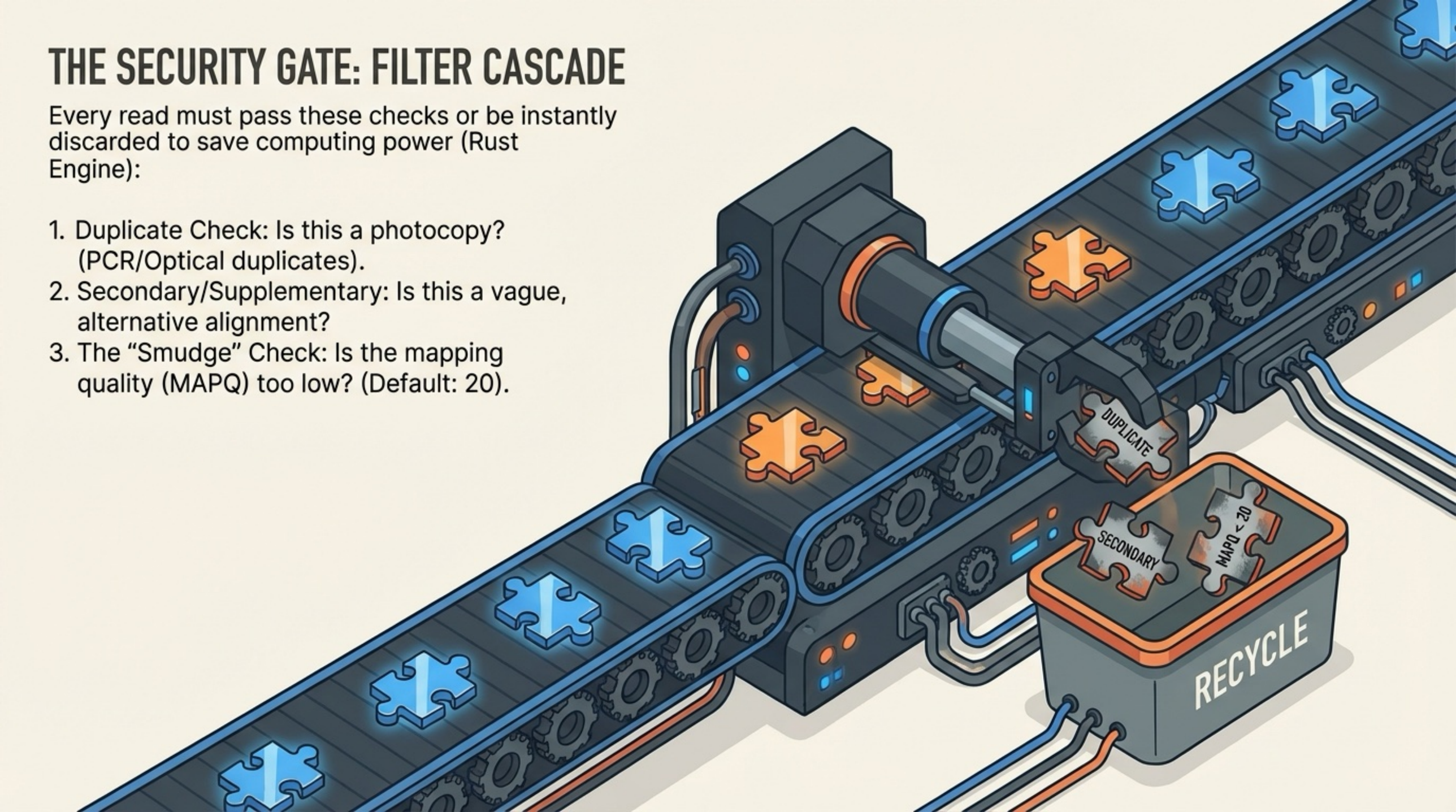
Why? Alignment Jitter. In repetitive regions, the evidence might be offset by a few bases. If we only grabbed the exact center, we would miss the evidence hiding just millimeters to the left.



# THE SECURITY GATE: FILTER CASCADE

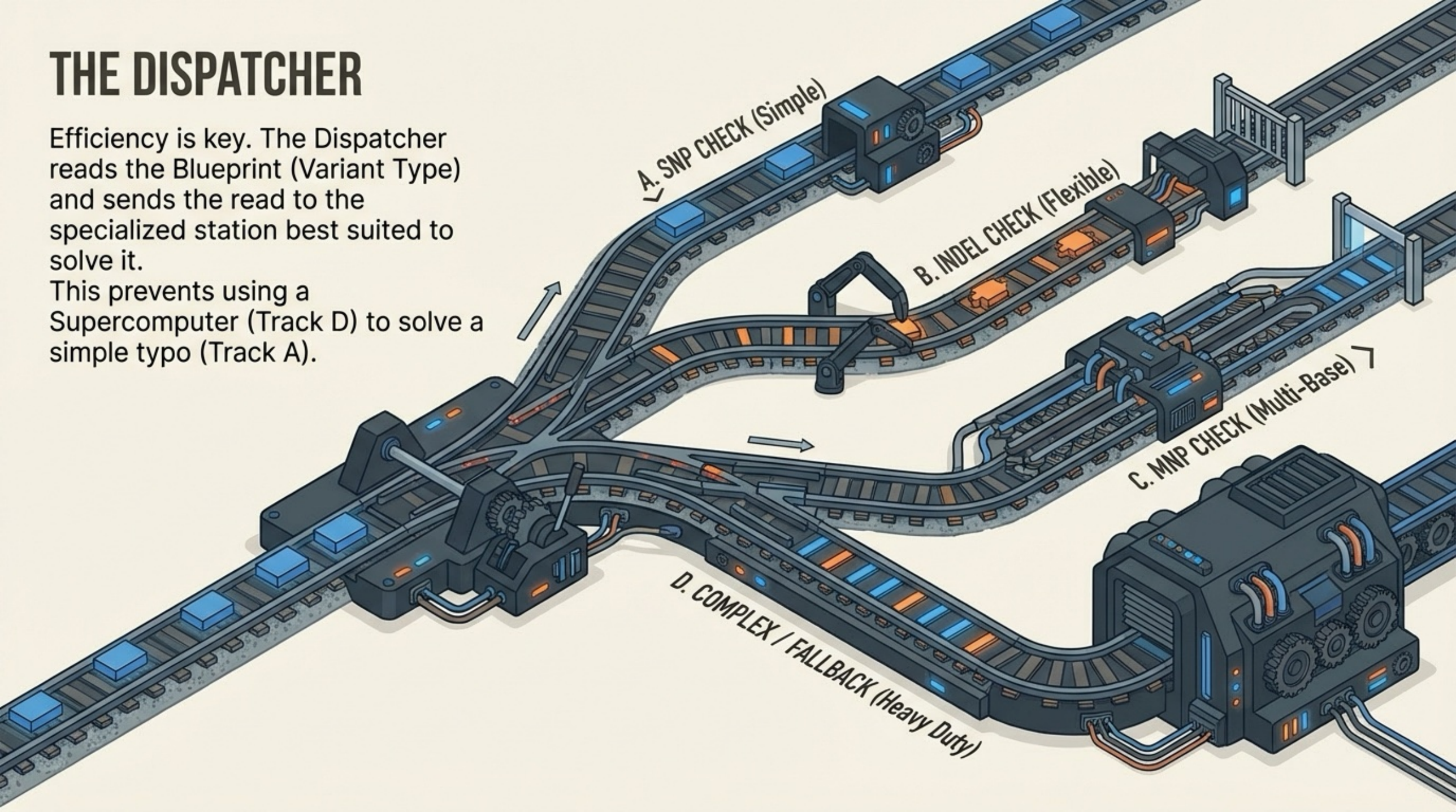
Every read must pass these checks or be instantly discarded to save computing power (Rust Engine):

1. Duplicate Check: Is this a photocopy? (PCR/Optical duplicates).
2. Secondary/Supplementary: Is this a vague, alternative alignment?
3. The “Smudge” Check: Is the mapping quality (MAPQ) too low? (Default: 20).



# THE DISPATCHER

Efficiency is key. The Dispatcher reads the Blueprint (Variant Type) and sends the read to the specialized station best suited to solve it.
This prevents using a Supercomputer (Track D) to solve a simple typo (Track A).

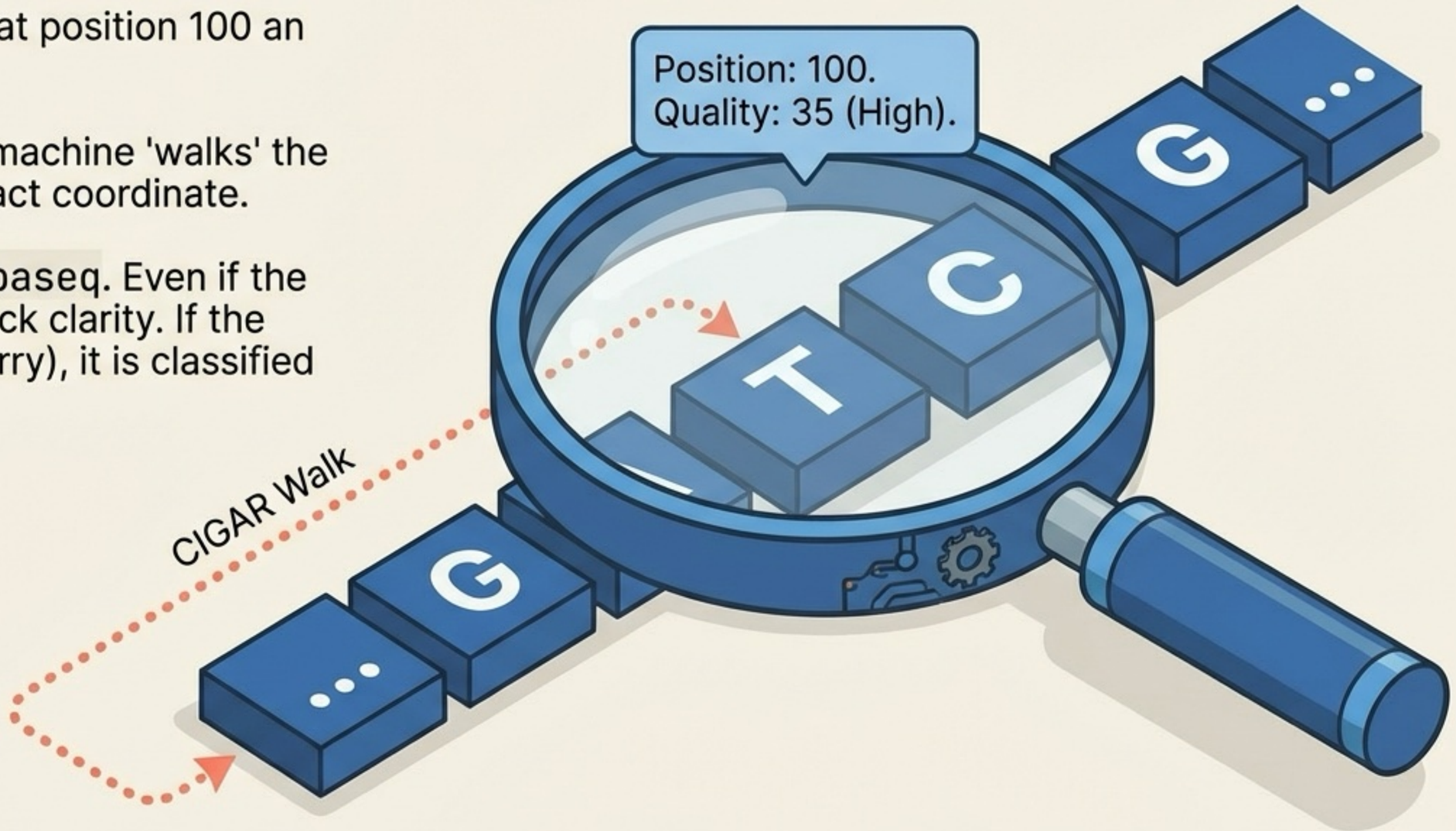


# STATION A: THE SNP CHECKER

**The Task:** Is the letter at position 100 an 'A' or a 'T'?

**The Mechanism:** The machine 'walks' the CIGAR string to the exact coordinate.

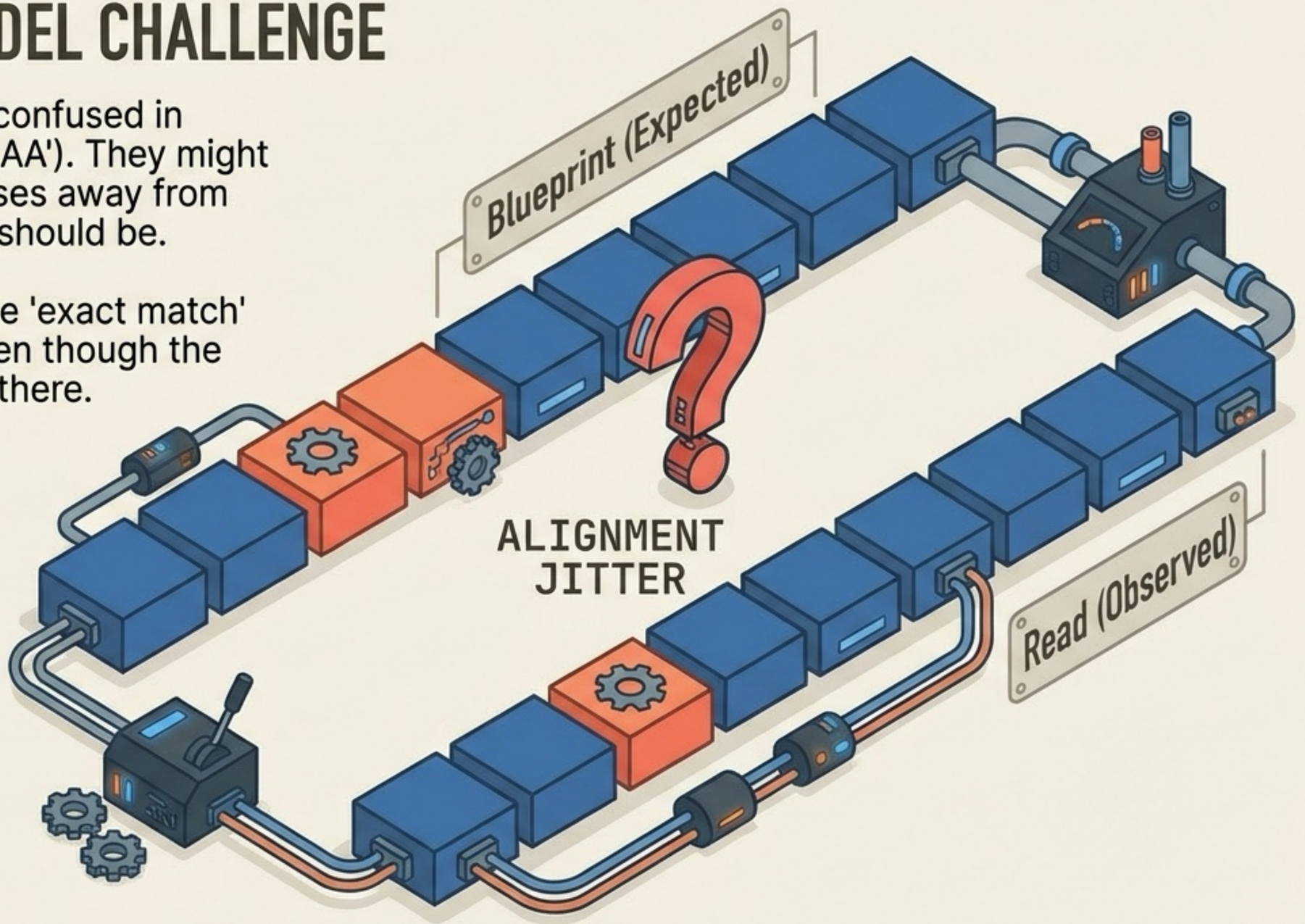
**Quality Control:** `min_baseq`. Even if the letter matches, we check clarity. If the base quality is low (blurry), it is classified as 'Neither/Unknown'.



# STATION B: THE INDEL CHALLENGE

**The Problem:** Aligners get confused in repetitive regions (like 'AAAAA'). They might place an insertion a few bases away from where the Blueprint says it should be.

**The Consequence:** A simple 'exact match' match' check would fail, even though the biological evidence is right there.
We need a flexible tool.

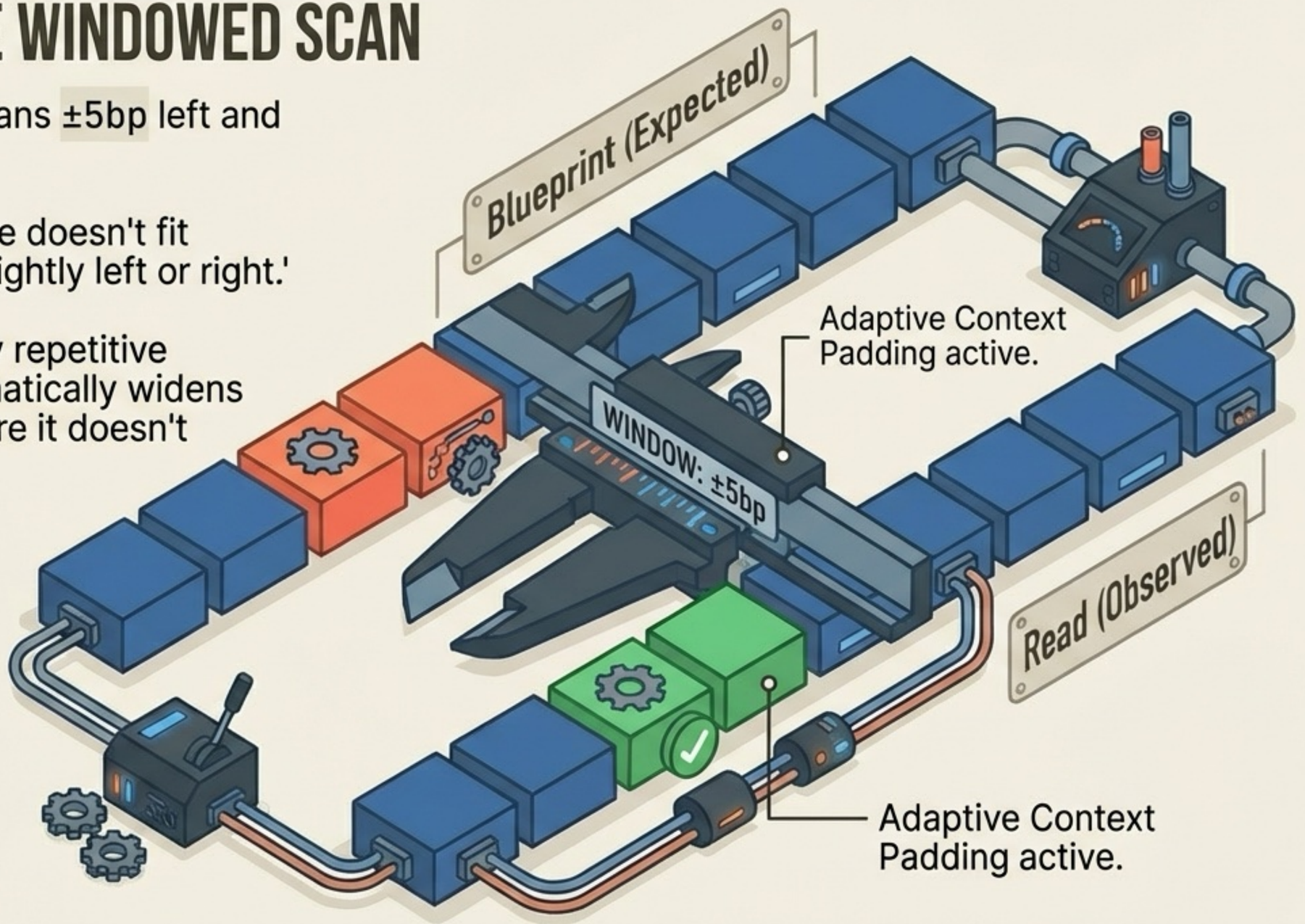


# THE SOLUTION: THE WINDOWED SCAN

**The Logic:** The machine scans ±5bp left and right of the anchor.

**Analogy:** 'If the puzzle piece doesn't fit exactly here, try sliding it slightly left or right.'

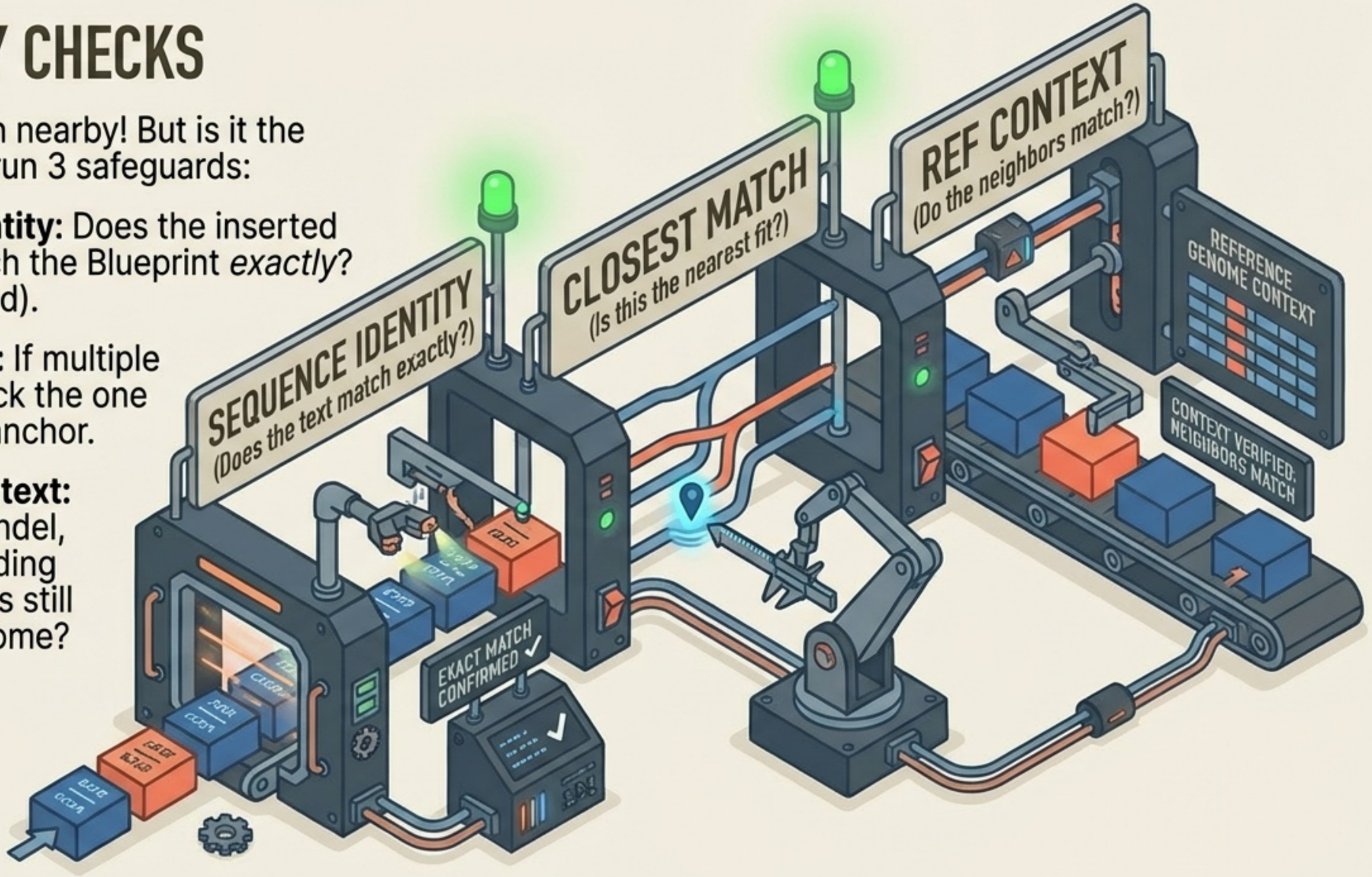
**Adaptive Context:** In highly repetitive regions, the machine automatically widens this search window to ensure it doesn't get lost in the pattern.



# THE SAFETY CHECKS

We found a match nearby! But is it the *right* match? We run 3 safeguards:

1. **Sequence Identity:** Does the inserted sequence match the Blueprint *exactly*? (Quality-masked).
2. **Closest Match:** If multiple spots fit, we pick the one closest to the anchor.
3. **Reference Context:** If we shift the indel, do the surrounding reference bases still match the genome?

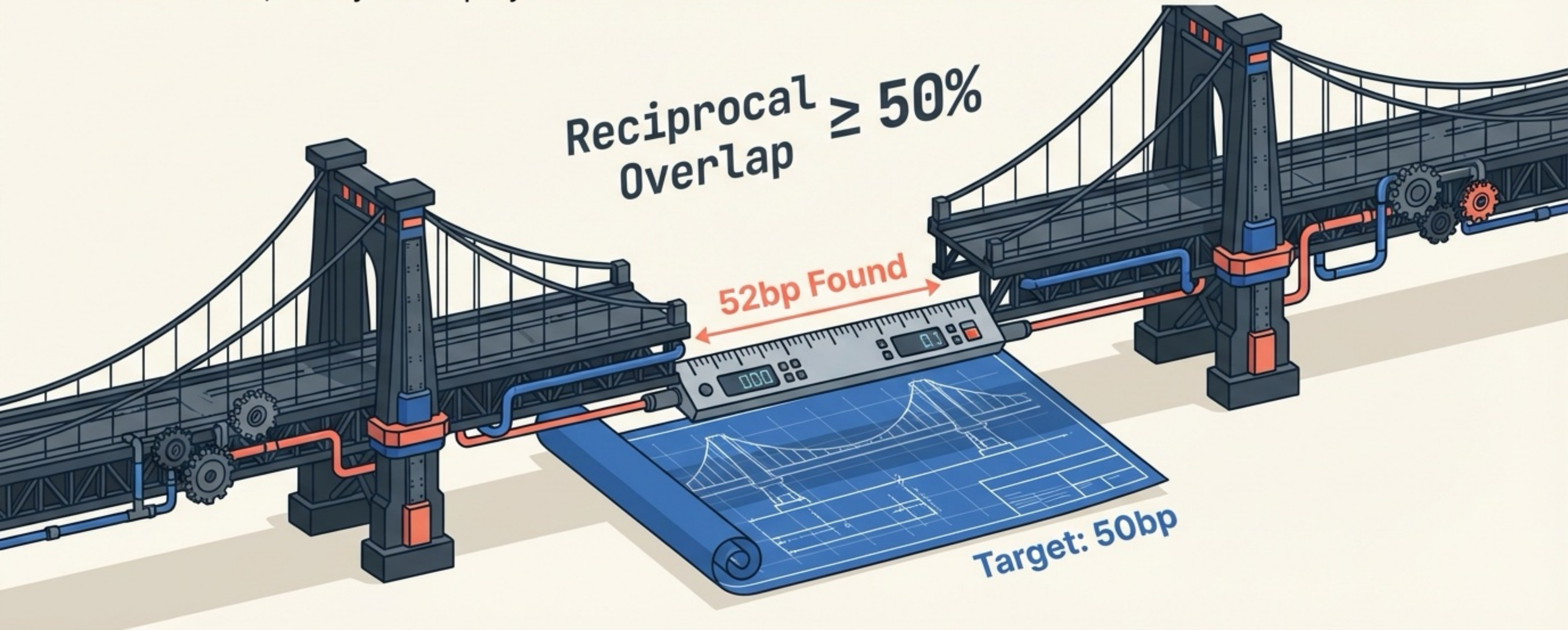


# SPECIAL HANDLING: THE GIANT DELETION

**The Scenario:** Large deletions (>50bp). **The Rule:** Reciprocal Overlap.

**The Logic:** Sometimes the deletion found is 52bp long, but the blueprint says 50bp. Do we count it?

**The Math:** Yes, if they overlap by at least **50%**. This mimics structural variant callers like SURVIVOR.

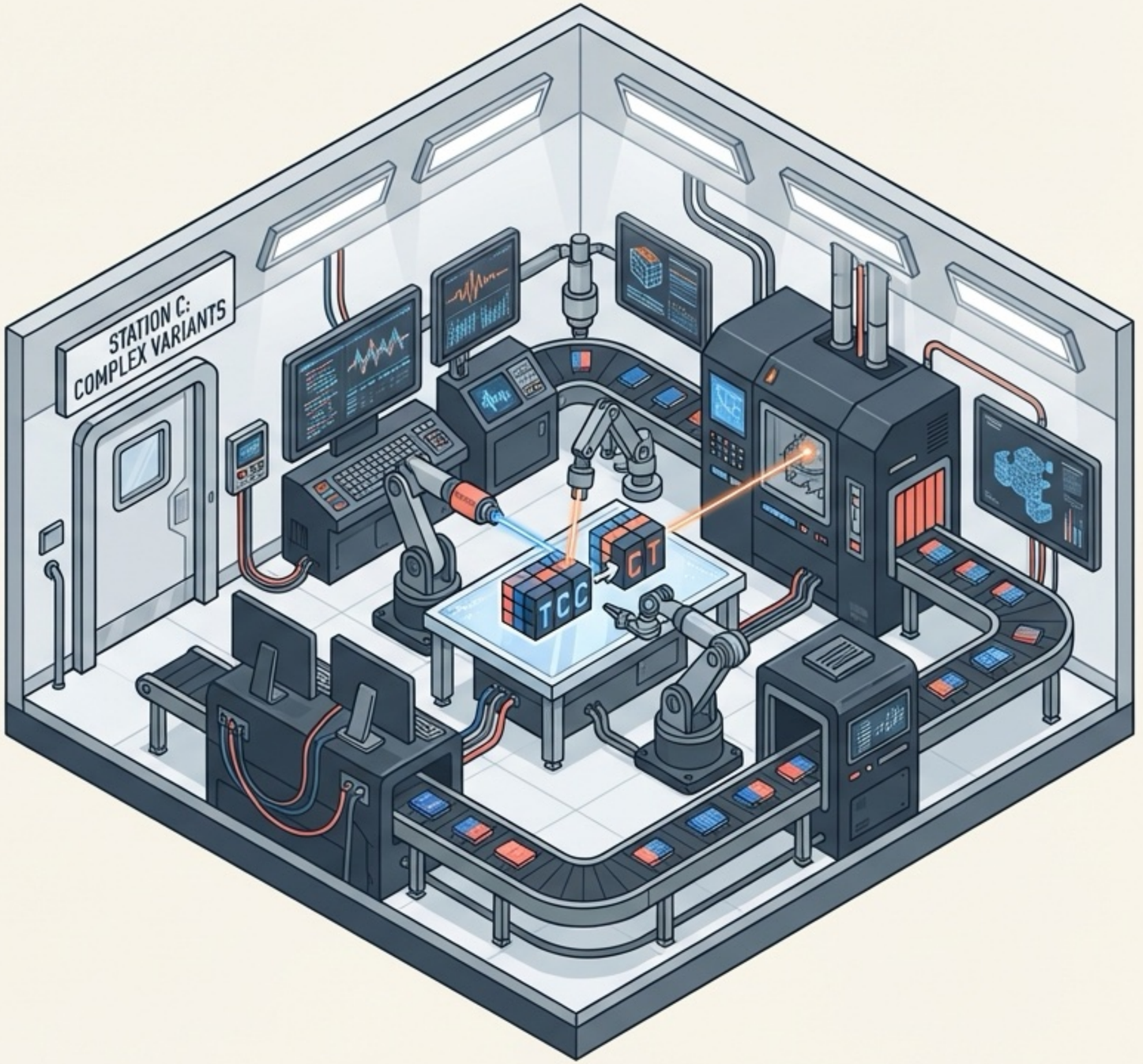


# STATION C:
# THE MASTER SOLVER

**The Target:** Complex variants where length and sequence change simultaneously (e.g., `TCC` → `CT`).

**The Approach:** The simple tools (Station A and B) won't work. We need a 3-Phase approach to deconstruct and rebuild the read.

*Note:* This is also the 'Fallback' station if Station A or B get confused.

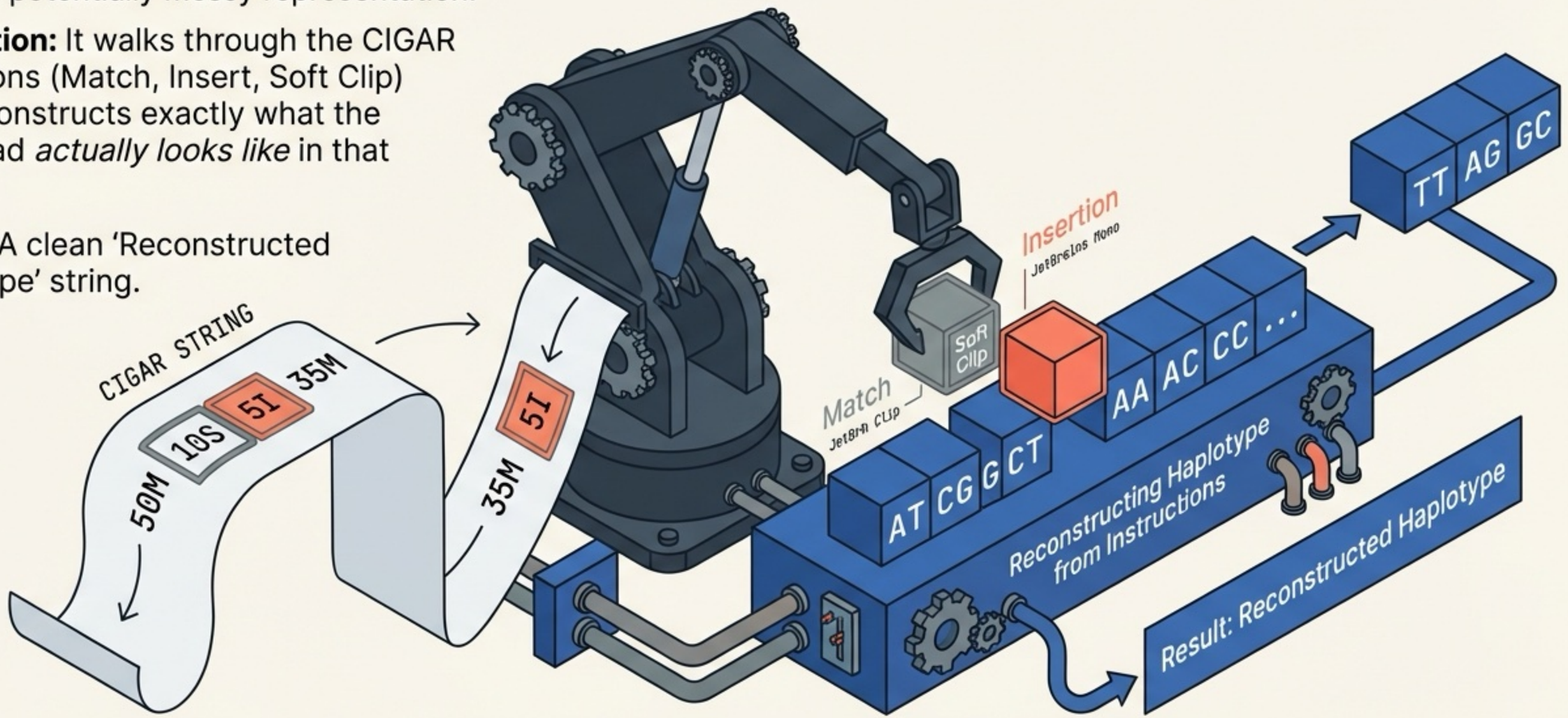


# PHASE 1: HAPLOTYPE RECONSTRUCTION

**The Concept:** The machine ignores the aligner's potentially messy representation.

**The Action:** It walks through the CIGAR operations (Match, Insert, Soft Clip) and reconstructs exactly what the DNA read *actually looks like* in that region.

**Result:** A clean 'Reconstructed Haplotype' string.



# PHASE 2: THE MASKED MATCH

**The Logic:** Compare the Reconstructed Haplotype against the Blueprint (ALT) and Standard (REF).

**The Mask:** Any base with quality `< min_baseq` is ignored. We don't let bad data vote.

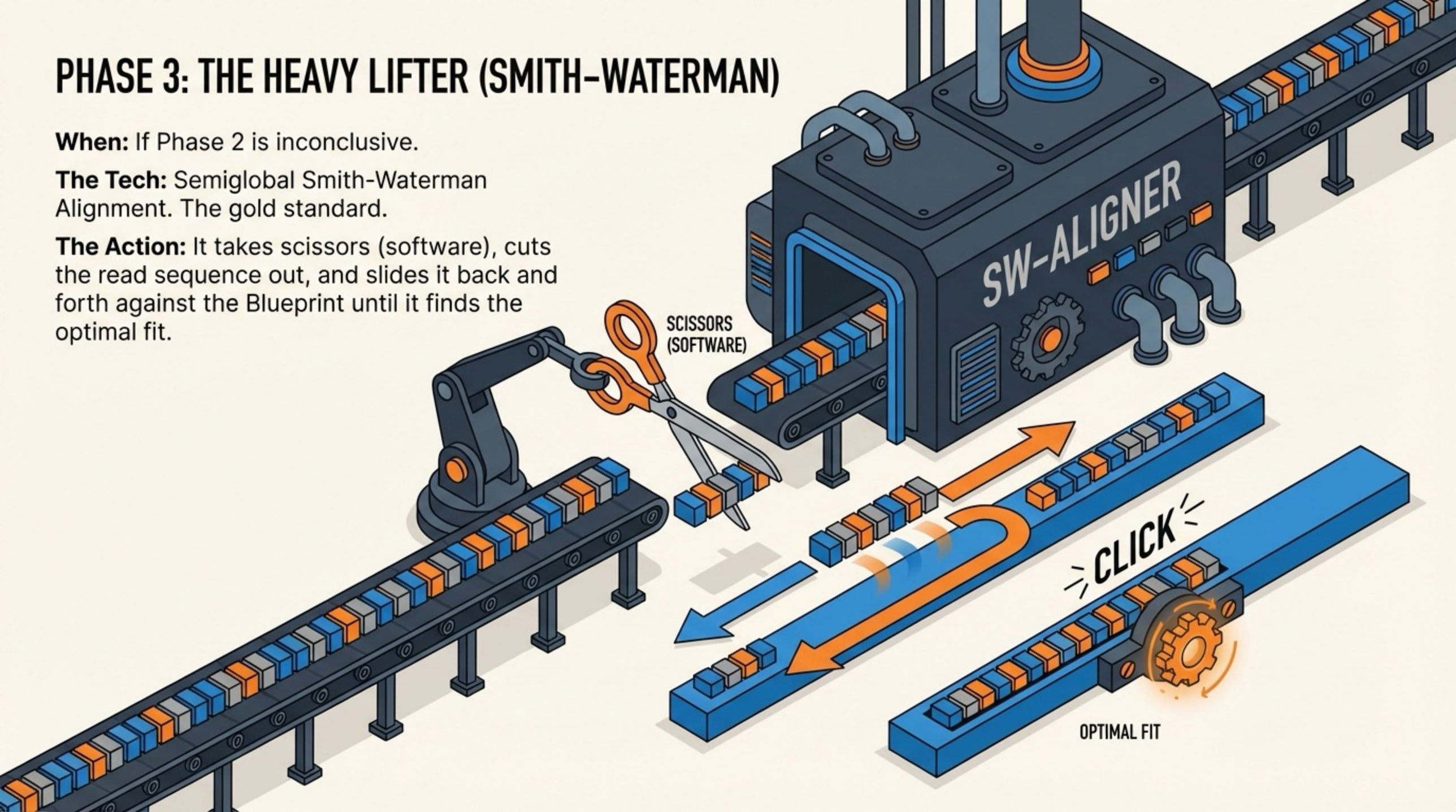
**Ambiguity Detection:** If the clear bases match *both* REF and ALT, we discard the read as 'Ambiguous' rather than guessing.

# PHASE 3: THE HEAVY LIFTER (SMITH-WATERMAN)

**When:** If Phase 2 is inconclusive.

**The Tech:** Semiglobal Smith-Waterman Alignment. The gold standard.

**The Action:** It takes scissors (software), cuts the read sequence out, and slides it back and forth against the Blueprint until it finds the optimal fit.

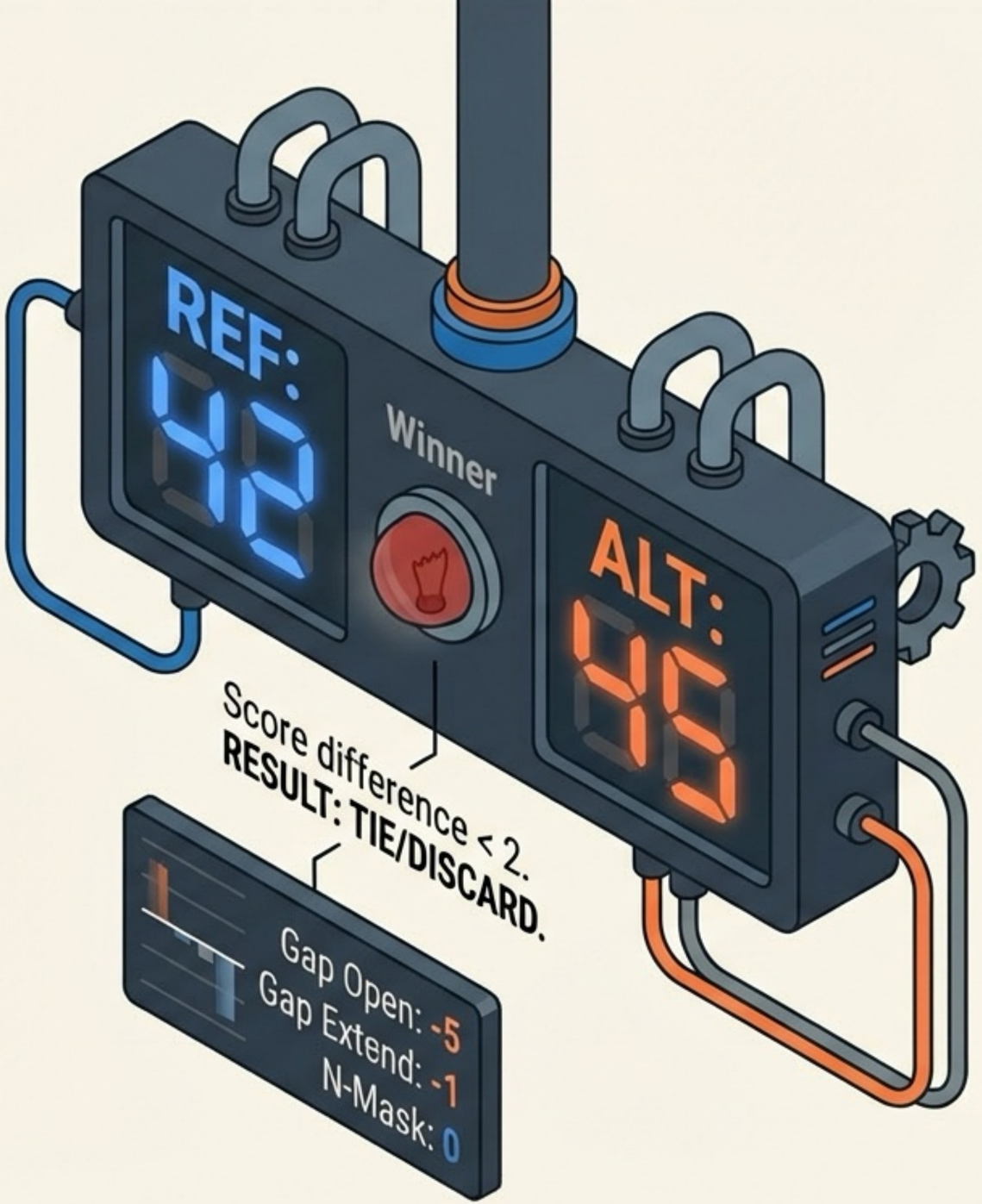


# PHASE 3: THE SCOREBOARD

**Gap Penalties:** It costs points to open a gap (insertion/deletion).

**N-Masking:** Low-quality bases are turned into 'N's (wildcards) that score 0.

**The Margin:** To declare a winner, the ALT score must beat the REF score by a margin of **≥ 2 points**. If it's too close, the read is discarded.



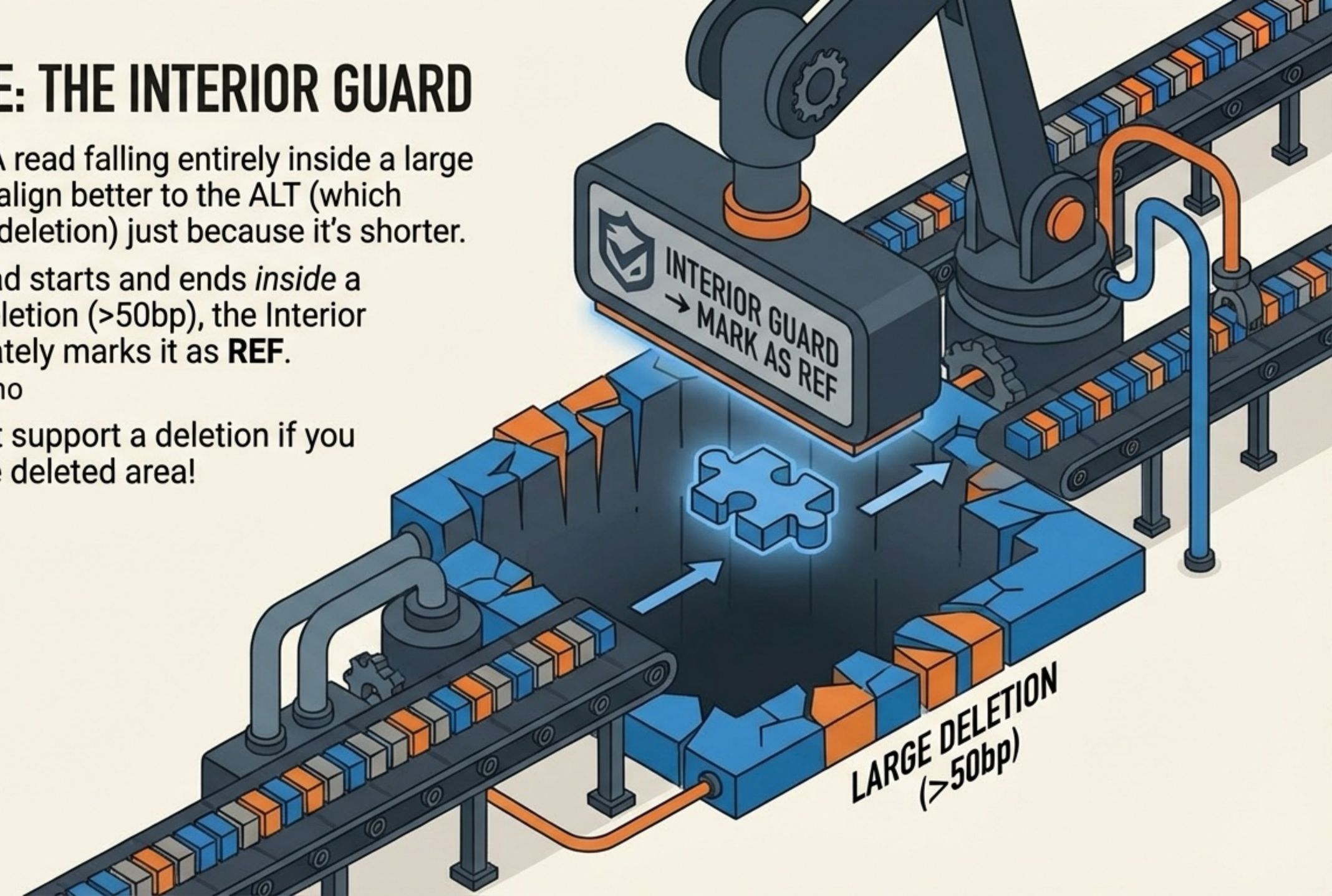
# EDGE CASE: THE INTERIOR GUARD

**The Problem:** A read falling entirely inside a large deletion might align better to the ALT (which represents the deletion) just because it's shorter.

**The Fix:** If a read starts and ends *inside* a known large deletion (>50bp), the Interior Guard immediately marks it as **REF**.

`JetBrains Mono`

**Why?** You can't support a deletion if you exist inside the deleted area!

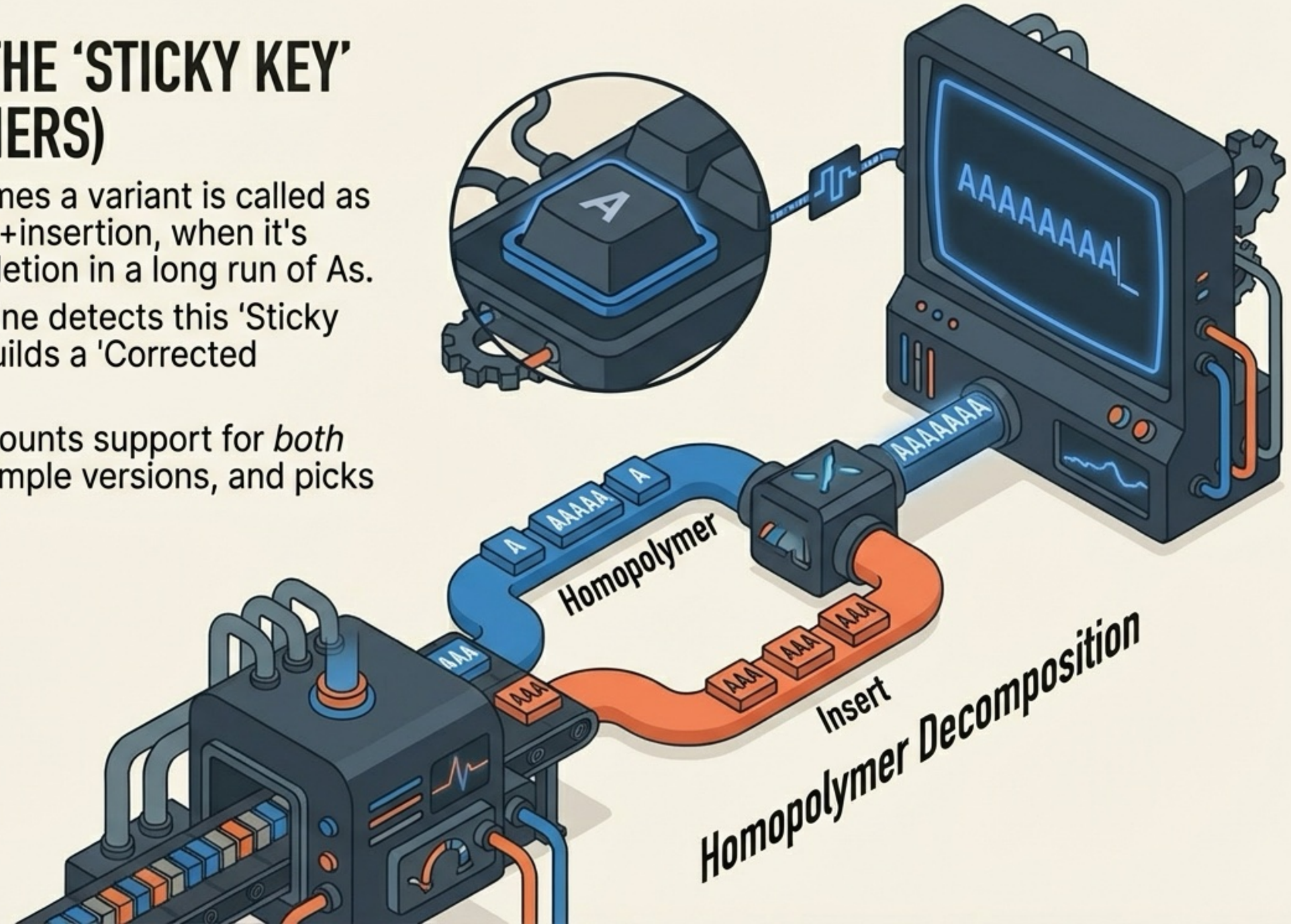


# EDGE CASE: THE ‘STICKY KEY’ (HOMOPOLYMERS)

**The Issue:** Sometimes a variant is called as a complex deletion+insertion, when it's really just a 1bp deletion in a long run of As.

**The Fix:** The machine detects this ‘Sticky Key’ pattern and builds a 'Corrected Blueprint'.

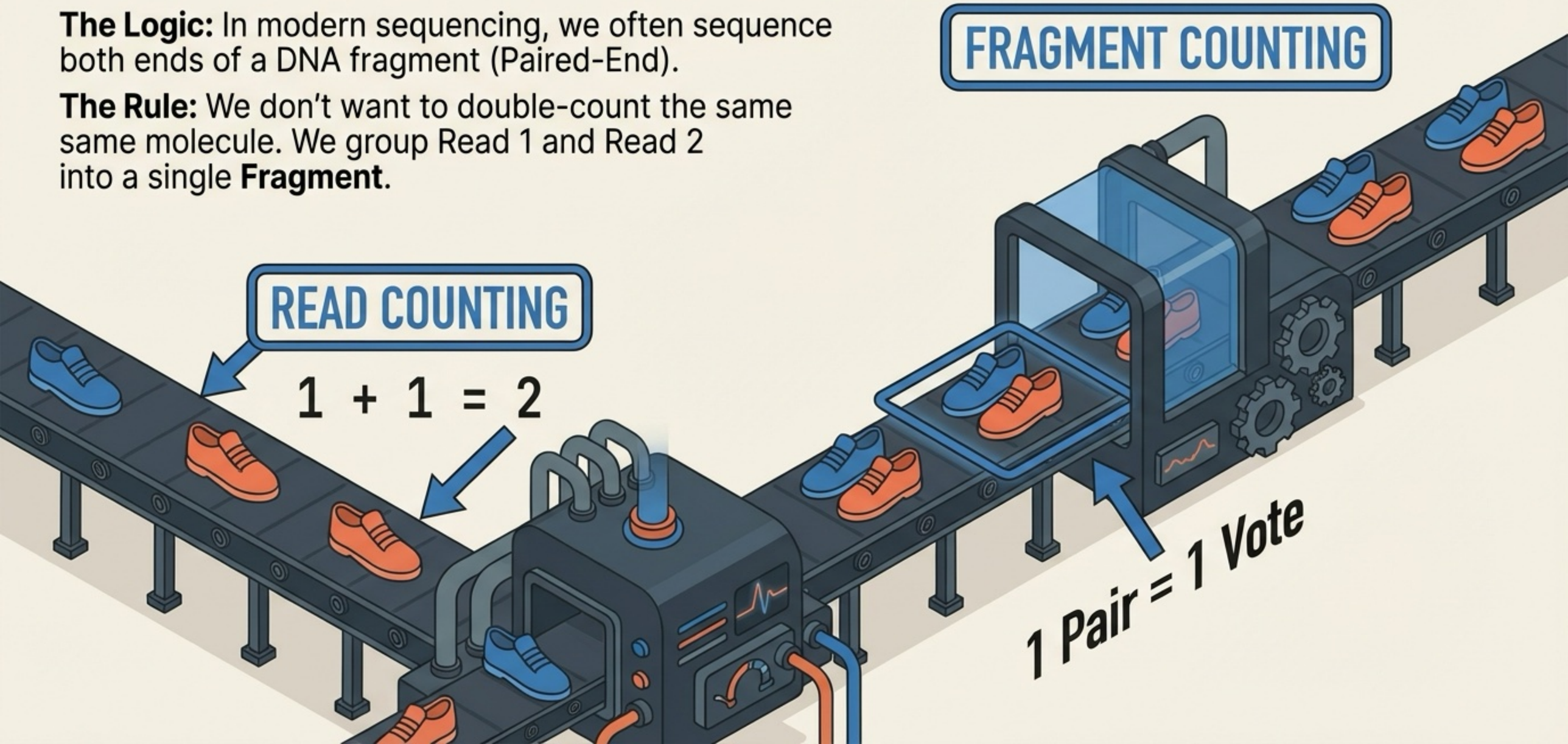
**Dual Counting:** It counts support for *both* the complex and simple versions, and picks the winner.



# COUNTING: READS VS. FRAGMENTS

**The Logic:** In modern sequencing, we often sequence both ends of a DNA fragment (Paired-End).

**The Rule:** We don't want to double-count the same same molecule. We group Read 1 and Read 2 into a single **Fragment**.

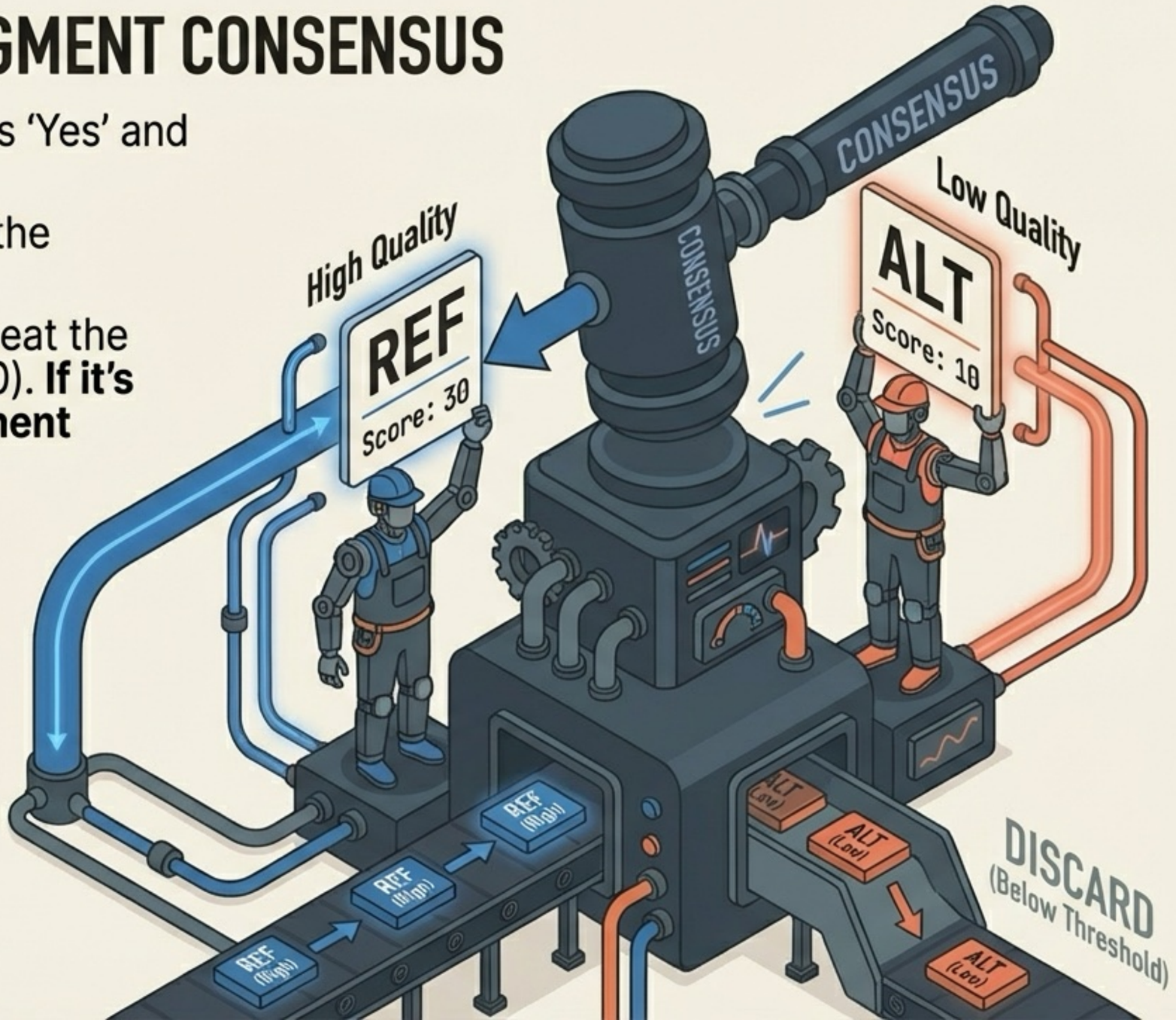


# THE TIE-BREAKER: FRAGMENT CONSENSUS

**Conflict:** What if the Left Read says ‘Yes’ and the Right Read says ‘No’?

**Resolution:** We trust the one with the clearer picture (higher quality).

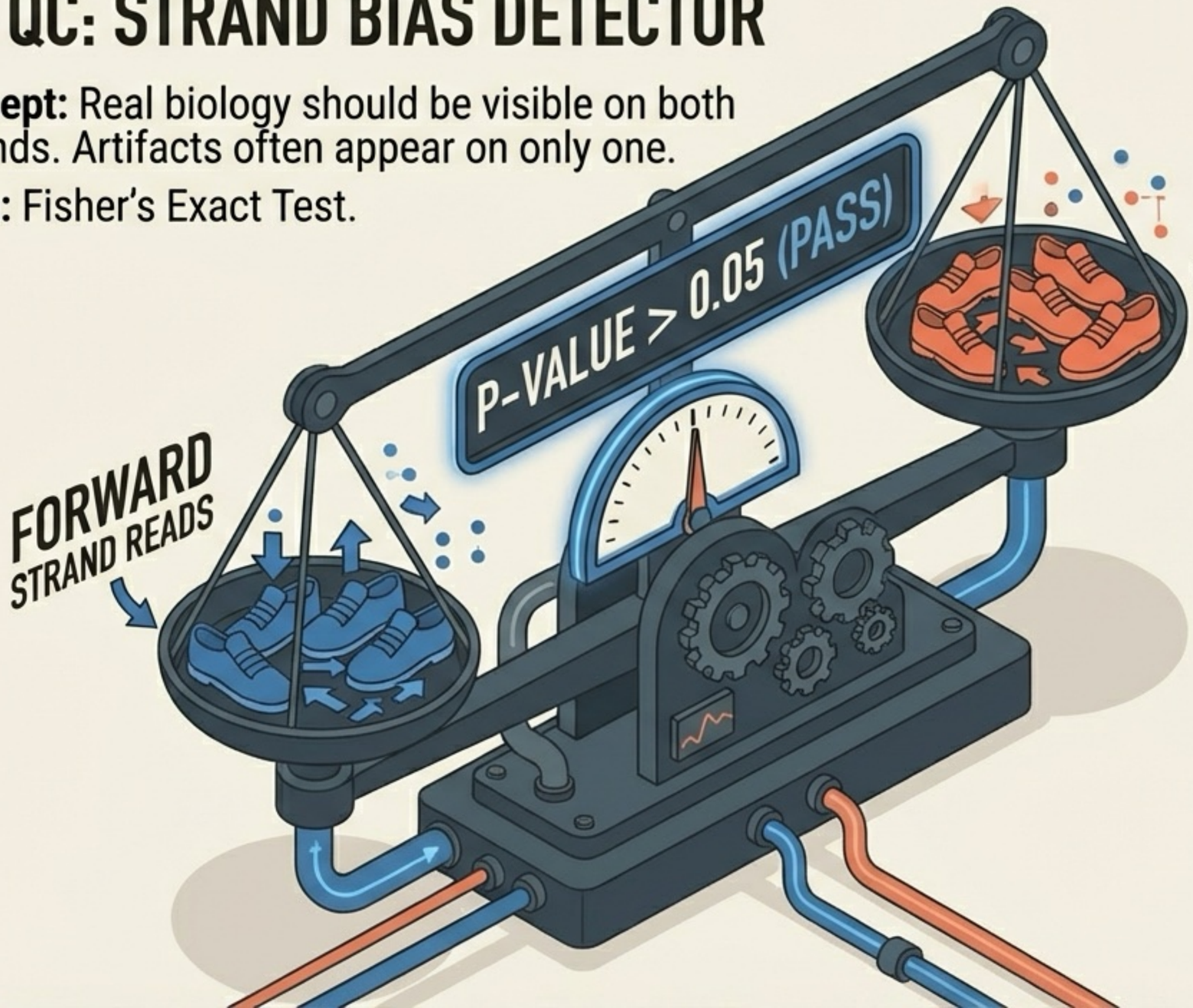
**The Threshold:** The winner must beat the loser by a quality margin (default 10). **If it’s close, we discard the whole fragment to be safe.**



# FINAL QC: STRAND BIAS DETECTOR

**The Concept:** Real biology should be visible on both DNA strands. Artifacts often appear on only one.

**The Math:** Fisher's Exact Test.





**THE RESULT:** If 100% of your variants are on the **Forward strand**, the machine flags a "Strand Bias" warning.

# THE COMPLETE CIRCUIT

py-gbcms transforms chaotic raw data into precise, statistically validated variant counts. It turns the mechanical ‘how’ into biological ‘what’.