BEYOND PASS RATES: ATTRIBUTING RELATIVE-SPATIAL FAILURES IN GENERATED ANIMATION CODE

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models are increasingly utilized to generate animation programs for explanatory videos, yet relative-spatial instructions remain substantially harder than absolute motion or temporal constraints. Existing work establishes this performance gap but largely treats spatial failure as a single phenomenon and evaluates interventions through aggregate pass rates. In this paper, we introduce REFPROBE, a diagnostic framework that attributes relative-spatial failures to distinct underlying causes using two signals exposed by program generation: (i) the observed placement behavior in the execution trace; (ii) the geometric quantities consulted by the generated program. Our framework assigns failures under a fixed decision procedure and enables interventions to be analyzed by the causes they repair and the behaviors they regress, rather than only by their net effect on pass rate. Across seven open-weight generators and 42,000 generated programs, we find that relative-spatial failures are strongly heterogeneous across both causes and models, with a substantial fraction arising from object-extent errors, reference anchoring failures, and verifier-boundary mismatches. Moreover, interventions with modest aggregate effects often produce large but opposing repair and regression effects at the mechanism level. These results suggest that aggregate pass rates are insufficient for diagnosing spatial generation failures and motivate cause-aware evaluation of program-generating models.

1 INTRODUCTION

Language models are increasingly used to generate animation programs for explanatory videos and motion graphics. Yet relative-spatial instructions remain substantially harder than absolute motion or temporal ordering. Two independent benchmarks report the same pattern: models perform well on simple motion and timing constraints, but degrade sharply when one object must be placed relative to another (Ma & Agrawala, 2025; Zhang et al., 2026). This gap has been observed at different scales and with different evaluation pipelines. The phenomenon is therefore well established. What remains unclear is why these failures occur.

Existing work largely treats relative-spatial failure as a single category. Proposed remedies include layout planning (Li et al., 2026c), spatial prompting (Premsri & Kordjamshidi, 2025), geometric constraints (Zhang et al., 2026), and self-consistent resampling (Ma et al., 2026). These methods are useful engineering solutions, but they are typically evaluated through a single aggregate pass rate, which is unable to answer what kind of failure they fix. Small changes in pass rate may indicate either an intervention is weak or the remedy produces large improvements on one subset of failures while simultaneously breaking another. These cases are indistinguishable without a causal diagnosis.

Generated programs provide an opportunity to go beyond this aggregate view. Unlike a rendered image alone, a program exposes both what the final placement did and how that placement was computed. From the execution trace, we can determine whether a placement violates a relation only near the verifier's decision boundary or whether it ignores object extent altogether. From the generated code, we can determine if the target position was indeed computed from the referenced object. These distinctions correspond to different mechanisms suggesting varied remedies, although producing visually similar failures.

To utilize this opportunity of generated programs, we introduce REFPROBE, a diagnostic framework for relative-spatial failure in generated animation code. REFPROBE combines two complementary

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

views of each failure: a *behaviour* view derived from the execution trace and a *provenance* view derived from the quantities used by the generated program. We then attribute animation program generation failures to four categories under a fixed decision procedure: failure to anchor on the referent, failure to account for object extent, disagreement with the verifier's spatial convention, and an explicit residual category. The decision procedure is further validated via fault injection and counterfactual tests. Fault injection constructs manifested error to probe if the procedure is able to detect the root cause, while counterfactual tests perturb the scene and check whether the target follows the referent.

Based on this framework, we study seven open-weight generators across 42,000 generated programs. Our analysis yields four main findings: (i) Generated programs are almost always syntactically valid and executable, yet they frequently implement the wrong spatial relation. (ii) Models often articulate the correct spatial constraint but fail to translate it into correct code, particularly for compositionally complex instructions. (iii) Relative-spatial failures do not arise from a single mechanism. Their attributed causes are highly non-uniform, vary substantially across models, and shift with model scale. (iv) Cause-level analysis reveals that instruction-level interventions can repair many previously failing cases while simultaneously causing regressions on cases that previously succeeded. These opposing effects may largely cancel out in aggregate pass rates, obscuring how an intervention actually changes model behavior. These findings show that aggregate pass rates obscure both why generated programs fail and how interventions alter their behavior, motivating cause-aware evaluation of program-generating models.

This work is diagnostic: it provides a criterion for asking what a relative-spatial failure is, validates that criterion, measures how its causes vary across models and evaluation conventions, and shows why these distinctions matter when interventions are judged.

Our main contributions are summarized as:

- **C1. A cause-level diagnostic framework.** We introduce REFPROBE, attributing relative-spatial failures using two signals available in generated programs: observed placement behaviour from the execution trace and parameter provenance from the code. The framework distinguishes failures due to reference anchoring, object extent, and verifier convention, while retaining an explicit residual category.
- **C2. A validated empirical decomposition of spatial failure.** We validate the attribution procedure with fault injection and an independent counterfactual provenance test, and apply it across seven open-weight generators and 42,000 programs. The resulting failure distribution is heterogeneous, model-dependent, and sensitive to the verifier's tolerance convention rather than being well described by a single spatial-error category.
- **C3. Cause-aware evaluation of interventions.** We evaluate four paired instruction-level probes and show that aggregate pass rates can hide large and opposing repair and regression effects. Cause-level analysis reveals which failures an intervention actually fixes, which successful cases it breaks, and why an apparently reasonable intervention can backfire.

2 RELATED WORK

2.1 FROM SPATIAL FAILURE TO MECHANISM ATTRIBUTION

Motion verification for animation code reveals a pronounced gap between motion types: over 5,600 prompts, first-attempt pass rates are 88.3% for single atomic motion and 88.5% for temporally relative motion, but only 37.9% for spatially relative motion (Ma & Agrawala, 2025); a larger programmatic benchmark reports the same pattern (Zhang et al., 2026). Yet neither study identifies the underlying cause, focusing instead on descriptive error modes or instruction-independent geometric checks such as overlap and out-of-bounds placement. One possible explanation is a gap between understanding a spatial principle and applying it correctly, consistent with prior work on comprehension without competence (Zhang, 2025) and spatial frames of reference (Premsri & Kordjamshidi, 2025; Levinson, 2003; Talmy, 1983). Existing interventions, including layout planning, spatial prompting, geometric constraints, and self-consistent resampling (Li et al., 2026c; Premsri & Kordjamshidi, 2025; Zhang et al., 2026; Ma et al., 2026), mainly report aggregate performance gains without determining which

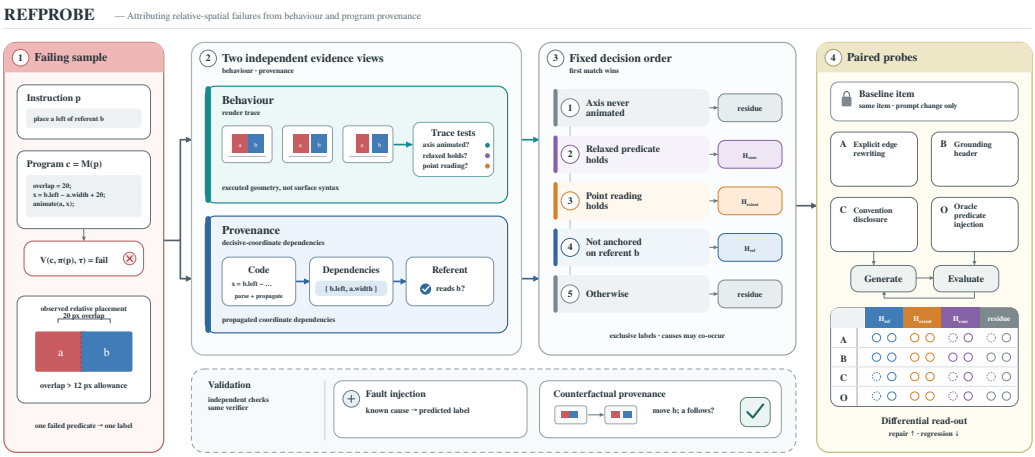


Figure 1: REFPROBE. A failing sample enters a discriminator reading two independent things about it: what the *placement* did, from the render trace, and what the *program* consulted, from parameter provenance.

mechanism produced the failure. We therefore treat the spatial gap as established and focus on decomposing its underlying causes.

2.2 SPATIAL REASONING BEYOND ANIMATION AND PROGRAM-BASED EVALUATION

The same weakness appears well beyond animation code. Text-only and multimodal benchmarks show that models struggle with relative spatial language (Mirzaee et al., 2021; Shi et al., 2022; Liu et al., 2022; Kamath et al., 2023; Wang et al., 2024), although these benchmarks evaluate understanding rather than emitted programs. Video-generation benchmarks independently identify dynamic spatial relations as a major failure mode (Liu et al., 2023b; Sun et al., 2024; Huang et al., 2024; Zheng et al., 2025; Liu et al., 2023a), often attributing the problem to text-encoder capacity or the difficulty of steering diffusion. Similarly, Xing et al. (2024) associate geometric quality with numeric tokenisation. These explanations are tied to particular generation pipelines and therefore cannot directly explain a setting in which a model writes an executable program, yet the spatial weakness persists there as well. Our study isolates this latter route: models that emit executable programs. Executing generated code and checking its behaviour provides the methodological basis for that analysis. Program execution is standard in code-generation evaluation (Chen et al., 2021; Austin et al., 2021; Du et al., 2023; Jain et al., 2024; Jimenez et al., 2023), while visual-program benchmarks extend the same principle to charts, interfaces, and vector graphics (Wu et al., 2024; Yang et al., 2024; Si et al., 2024; Zou et al., 2024; Xing et al., 2023; Ku et al., 2025). Animation-code systems likewise increasingly use programs as the representational carrier (Chen et al., 2026; Li et al., 2026a;b; Shi et al., 2026). Repair and self-correction methods require a signal about *what* went wrong (Chen et al., 2023; Madaan et al., 2023; Shinn et al., 2023; Chen et al., 2022). Our framework combines execution behaviour with program provenance to diagnose relative-spatial failures and analyse how their attributed causes change under different prompting interventions.

3 REFPROBE

3.1 PROBLEM FORMULATION

Let p be an animation instruction, $\pi(p)$ be its gold first-order predicate program, and M be a generator emitting $c = M(p)$. An evaluator V executes c, samples every object's geometry over the timeline, and decides $V(c, \pi(p), \tau) \in \{\mathsf{pass}, \mathsf{fail}\}$ under a numeric tolerance τ — the protocol motion verification already uses (Ma & Agrawala, 2025) and larger benchmarks instantiate at scale (Zhang et al., 2026). A *relative-spatial failure* is a triple $(p, \pi(p), c)$ with $V = \mathsf{fail}$ in which $\pi(p)$ constrains the position of a target a relative to a named referent b. Our study focuses on this set.

3.2 THREE HYPOTHESES

We split relative-spatial failure into three causes plus an explicit residue; Fig. 1 shows where each is decided. The three are made mutually exclusive by the decision order below, not by nature: a single failure can exhibit more than one, and §6 reports how often they co-occur.

H_{ref} **(reference-frame resolution).** The program does not anchor the target on the referent: the expression computing the decisive coordinate mentions no geometric quantity of b, so the placement is canvas-absolute however it was arrived at. This is *anchor choice*, related to but not the same as the frame-of-reference distinction of Levinson (2003) that Premsri & Kordjamshidi (2025) evaluate.

H_{extent} **(object extent ignored).** The placement is correct when both objects are treated as points and wrong when they are treated as boxes: "left of b" is realised as a centre comparison rather than an edge comparison.

H_{conv} **(predicate-convention mismatch).** The placement misses the relation only by the width of the verifier's tolerance: it fails the strict predicate and satisfies it under the pre-registered relaxed band. For separation relations, whose strict threshold is zero, that band is an allowance of 12 px of overlap. When this category accounts for a substantial fraction of failures, the reported failure rate partly reflects the verifier's tolerance convention. Standard evaluation counts these boundary-sensitive cases as model errors without distinguishing them from other spatial failures.

3.3 TWO AXES, ONE DECISION ORDER

Both axes are computable from artefacts the code route makes available, and neither is available to a pixel-level evaluator.

The **behaviour** axis is read from the render trace: a failure is *tolerance-only* when the predicate holds under the relaxed convention, and *extent-ignoring* when it holds under the *point reading* of the same relation (every bounding box replaced by its centre) but not under the relaxed convention. Alignment relations are extent-independent by construction and are never eligible for the second label.

The **provenance** axis is read from the program: we parse the emitted JavaScript, propagate local constants and variables to a fixpoint, and collect, for the last animation call targeting a, the scene quantities the decisive-coordinate expression depends on. A program is *anchored* when that set contains a geometric quantity of the referent b.

A behavioural axis is necessary because a purely syntactic criterion is not robust: models routinely constant-fold the referent's geometry into a literal, so a parser reports an expression that mentions nothing about the referent even though the placement tracks it. The decision order is fixed before any generation:

1. the decisive axis was never animated $\rightarrow$ *unclassified*;
2. the predicate holds under the relaxed tolerance $\rightarrow H_{\mathrm{conv}}$;
3. it holds under the point reading $\rightarrow H_{\mathrm{extent}}$;
4. the program is not anchored on the referent $\rightarrow H_{\mathrm{ref}}$;
5. otherwise $\rightarrow$ *unclassified*.

Each sample receives at most one attribution, based on the first relative-spatial predicate it fails. The ordering is a design choice with a measured footprint and we quantify its impact in §6. We also report the full behaviour $\times$ provenance contingency table to show how the two evidence axes interact and where program provenance provides diagnostic information beyond behaviour alone.

3.4 VALIDATING THE DISCRIMINATOR

Fault injection. We synthesise reference programs whose cause is known by construction: a hard-coded canvas coordinate (H_{ref}), a centre-only offset with a literal displacement (H_{extent}), and a full-geometry placement between the strict and relaxed thresholds (H_{conv}). We score the

discriminator against the injected label. Programs that still satisfy the strict predicate are excluded, being considered not to constitute failures.

Counterfactual provenance. A second criterion that never reads the code: we perturb the referent's geometry, re-execute the program, and check whether the target's final position changes accordingly. The two criteria share no parser and no provenance logic. Agreement between them therefore provides convergent evidence for the anchoring assessment. Both run on the same runtime and verifier, so it bounds analysis error rather than establishing that our three causes carve the phenomenon correctly.

3.5 Four Probes, Read Differentially

Every probe holds the scene, the gold predicate and decoding fixed, when comparing with the baseline and each other on the same item. The probes vary in the information provided in the prompt or the output requirements. ***A.*** **Explicit reference-frame rewriting** names the compared edges on the input side ("its right edge ends up at or to the left of the circle's left edge"). ***B.*** **Forced grounding header** makes the model emit `{anchor, anchor_quantity, target_quantity, relation, offset}` before the code. It's a modality transfer of the spatial-guided prompting of Premsri & Kordjamshidi (2025) and the one probe that changes the required output format rather than the instruction. ***C.*** **Convention disclosure** states the verifier's numeric thresholds for the predicates the item uses, the direct causal test of H_{conv}. ***O.*** **Oracle predicate injection** supplies the gold predicate program alongside a glossary of the predicate language. The read-out is the differential: which baseline causes a probe repairs *and* how many already-passing items it breaks. An intervention that repairs and regresses equally is invisible to a pass rate.

4 Experimental Setup

4.1 Carrier and Probe Set

Prior work uses a browser plus a timeline library and checks motion predicates against the rendered result (Ma & Agrawala, 2025; Zhang et al., 2026). We retain the shape and remove the browser. Programs are JavaScript against a four-function animation runtime that is deterministic and side-effect free: (`move`, `resize`, `rotate`, `fade`) . Its output is a per-frame trajectory of every object attribute. This design removes confound while preserving the quantities used to evaluate each predicate, which match those produced by a renderer. It also supports provenance analysis by declaring the geometric quantities available to the model as named constants in the scene prelude.

Each item ships a four-object scene, one instruction, and a gold predicate program in a 15-predicate first-order language over object geometry and animation timing. The probe set has 1000 items, 200 in each of five categories. Four of them follow the split of Ma & Agrawala (2025): *atomic* (one absolute motion, rotation or fade), *temporal* (two atomic motions plus an ordering constraint), *spatial* (one relative placement), and *spatio-temporal* (a relative placement plus a second, ordered animation). We construct the fifth *compound* tier: one object under two simultaneous relational constraints against two different referents, one decisive on each axis, so a second animation cannot repair the first. We added it after a pilot showed our carrier to be easier than the benchmarks the split comes from, and mark it as ours throughout. Appendix A gives the canvas geometry, the relation balance and the satisfiability guarantee every item carries.

4.2 Verifier and Tolerances

Two tolerance settings are frozen before any generation is run. The strict setting is what a benchmark would ship: separation relations require non-overlapping boxes, proximity, alignment and canvas-edge relations require a bounded pixel offset, and a temporal constraint means the later animation starts no earlier than the earlier one ends. The relaxed setting multiplies each threshold-bearing predicate by a pre-registered factor between $2\times$ and $5\times$ (median $3.5\times$) and allows 12 px of overlap on separation relations, which have a strict threshold of zero. Table 4 in the appendix gives every value. §5 reports how the cause mix moves when this whole band is halved or doubled. We report *pass@0* (one greedy sample, no retries) to evaluate the first attempt a deployed system would make.

Table 1: Baseline condition: first-attempt verification rate (pass@0, %) by instruction category, with 95% Wilson intervals. *Exec.* is the share of generated programs that run without error.

Model	Exec.	Atomic	Temporal	Spatial	Sp.-temp.	Compound	All
DS-Coder-6.7B	99.7	75.5 [69,81]	38.0 [32,45]	64.5 [58,71]	27.5 [22,34]	37.0 [31,44]	48.5
Olmo-3-7B	99.6	95.5 [92,98]	1.0 [0,4]	74.5 [68,80]	2.5 [1,6]	21.5 [16,28]	39.0
Qwen2.5-Coder-7B	100.0	99.5 [97,100]	94.5 [90,97]	80.5 [74,85]	68.5 [62,75]	53.5 [47,60]	79.3
Qwen2.5-7B	100.0	96.0 [92,98]	90.0 [85,93]	73.5 [67,79]	58.0 [51,65]	24.0 [19,30]	68.3
Qwen2.5-32B-AWQ	100.0	100.0 [98,100]	100.0 [98,100]	95.5 [92,98]	85.5 [80,90]	72.5 [66,78]	90.7
Qwen3-4B	100.0	98.0 [95,99]	92.0 [87,95]	90.0 [85,93]	69.5 [63,75]	41.0 [34,48]	78.1
Qwen3-8B	100.0	99.5 [97,100]	99.0 [96,100]	89.0 [84,93]	78.0 [72,83]	44.5 [38,51]	82.0

4.3 Conditions

Every model runs six conditions over the same 1000 items: **pred** (instruction → predicate program, no code), **code** (the baseline), and the four probes ***A***, ***B***, ***C***, ***O*** of §3. This is 6000 generations per model. The **pred** condition is scored two ways: *exact*, the model's predicate set equals the gold set after an order-invariant canonicalisation, and *core*, the gold set is contained in it. We use *core* as the comprehension measure, because models routinely emit the gold predicates plus supernumerary ones — on the compound tier, where the gold set has two predicates, the mean emitted set runs from 2.4 to 4.9 predicates across models — which exact matching would punish as a comprehension failure when the constraint was in fact stated.

4.4 Models and Serving

We evaluate seven open-weight instruction-tuned generators from four families over an $8\times$ parameter range (Table 5, appendix): Qwen2.5-7B-Instruct, Qwen2.5-Coder-7B-Instruct and Qwen2.5-32B-Instruct-AWQ (Qwen Team et al., 2024; Hui et al., 2024; Lin et al., 2023), Qwen3-4B and Qwen3-8B (Yang et al., 2025), DeepSeek-Coder-6.7B-Instruct (Guo et al., 2024) and Olmo-3-7B-Instruct, from the OLMo line (Team OLMo et al., 2025). All are served locally with vLLM (Kwon et al., 2023) under the fixed greedy configuration of Appendix D; 0.06% of generations reach the token budget overall and none in the baseline condition, so truncation is not a source of the failures we attribute. Decoding is deterministic, so every paired comparison differs only in the instruction.

4.5 Statistics

Shares carry Wilson intervals (Wilson, 1927); paired comparisons on the same prompts use an exact McNemar test over the discordant pairs (McNemar, 1947); distributional claims use χ^2 goodness-of-fit against uniformity and χ^2 independence for the two-axis table; discriminator agreement is Cohen's κ (Cohen, 1960).

5 Results

5.1 E1: The Probe Set and the Discriminator Are Valid

The reference solver satisfies the strict verifier on 1000 of 1000 items, so no item is unsatisfiable and no failure below is an artefact of an unreachable target (Table 3). Against programs whose cause is known by construction, the discriminator recovers the injected cause at precision 1.00 / 0.88 / 0.97 and recall 0.81 / 0.99 / 0.99 for H_{ref} / H_{extent} / H_{conv}; E5 is the complementary check on real model output. The one soft boundary is between H_{ref} and H_{extent} (Figure 3), since a hard-coded coordinate landing on the correct side of the referent while overlapping it is behaviourally identical to an extent error. The leak runs in one direction only, so our reported H_{ref} share is a conservative one.

5.2 E2: Difficulty Reproduction

Every model emits an executable program essentially always (99.6–100.0%), so nothing below is an execution failure: these programs run and place things wrongly. Table 1 and Figure 4 show the category ordering of prior work (Ma & Agrawala, 2025; Zhang et al., 2026) reproducing for the

Table 2: Open-weight generators and attribution of relative-spatial failures. All models are served locally with greedy decoding. n denotes the number of failing samples with a relational spatial predicate; attribution values report the share of failures (%). Par. is the parameter count, Prec. the serving precision, and Uncl. the unclassified residue.

Model Configuration				**Failure Attribution**				
Model	Family	Par.	Prec.	n	H_{ref}	H_{extent}	H_{conv}	Uncl.
DS-Coder-6.7B	DeepSeek	6.7B	bf16	279	10.8	30.8	25.4	33.0
Olmo-3-7B	OLMo	7.3B	bf16	265	24.9	35.1	16.6	23.4
Qwen2.5-Coder-7B	Qwen2.5	7.6B	bf16	192	9.4	42.2	28.6	19.8
Qwen2.5-7B	Qwen2.5	7.6B	bf16	265	8.7	61.9	20.4	9.1
Qwen2.5-32B-AWQ	Qwen2.5	32.8B	AWQ4	92	2.2	76.1	18.5	3.3
Qwen3-4B	Qwen3	4.0B	bf16	193	33.2	40.9	20.7	5.2
Qwen3-8B	Qwen3	8.2B	bf16	172	14.0	43.0	33.7	9.3
Pooled				1,458	15.6	44.4	23.3	16.8

five Qwen models: atomic motion is nearly solved (96.0–100.0%), temporal ordering close behind (90.0–100.0%), and relative-spatial placement markedly harder (73.5–95.5%). Those rates exceed the 37.9% Ma & Agrawala (2025) report for a much stronger closed model, because our carrier hands the generator exact scene geometry and asks for one relation at a time; the comparison to make is the ordering, not the level. The compound tier closes that distance: two simultaneous constraints drop pass@0 to 21.5–72.5% while changing nothing about the carrier, API or verifier, and the six sub-32B models land at 21.5–53.5%, around the 37.9% prior work reports — difficulty is compositional rather than lexical. The two non-Qwen models break the ordering rather than the level (Olmo-3-7B passes 95.5% of atomic instructions and 1.0% of temporal ones), so for them the binding constraint is composition, not space — which is why we report attribution per model.

5.3 E3: Models Understand the Constraint but Fail to Implement It

Table 6 and Fig. 2 compare two abilities on the same prompt: whether a model can produce the correct spatial predicate and whether it can generate code that satisfies that predicate. For atomic instructions, the two are similar for six of the seven models, with gaps of only 0.0–4.0 points (DeepSeek-Coder-6.7B is the exception at 24.5 points). On the compound tier, however, the gap becomes large for every model, ranging from 26.5 to 78.0 points. In other words, a model may correctly produce `left_of(a,b)` but still generate arithmetic that violates the relation. This shows that the failure is not simply due to misunderstanding the instruction: on many hard examples, the model identifies the correct constraint but fails to implement it in code.

The spatio-temporal setting reveals a different failure mode. For four of the five Qwen models, predicate performance is lower than code performance by 1.0–58.0 points. When asked only to translate a mixed spatial-and-temporal instruction into predicates, these models sometimes replace the relative relation with an absolute movement, for example producing `move_by(circle,−100,0)` instead of `left_of(circle,triangle)`. We refer to this behavior as *premature grounding*. Pooled across models, it occurs on 14.6% [13.5, 15.8] of items containing a relational spatial predicate (Table 7a). Thus, reference anchoring can fail even before code generation, during the intermediate representation of the instruction.

5.4 E4: Relative-Spatial Failures Have Different Causes

The attributed failures are not evenly distributed across the three causes (Table 2; Figure 5a, appendix). Among 1458 failures, 44.4% are attributed to H_{extent}, 23.3% to H_{conv}, and 15.6% to H_{ref}, with 16.8% left unclassified. The distribution differs significantly from uniform ($\chi^2(2) = 234.0$, $p < 0.0001$). Most of the unclassified cases also have a clear pattern: 198 of 245 never animate the decisive axis.

The share of H_{conv} depends on the verifier tolerance. Halving the relaxation band reduces its share to 13.2%, while doubling the band increases it to 36.5%. H_{extent} changes in the opposite direction, from 53.9% to 32.9%. This sensitivity is expected because H_{conv} is defined by the verifier boundary. Importantly, it remains a substantial source of failure across all tolerance settings we test. The overall

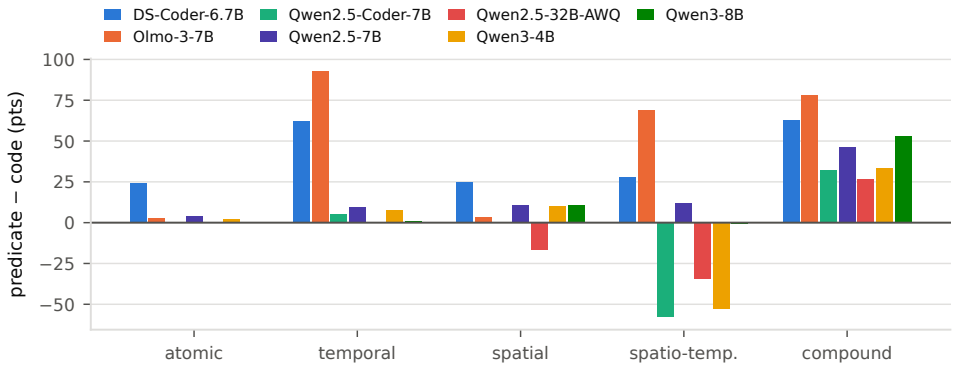


Figure 2: Paired comprehension–generation gap by category and model: predicate synthesis minus code pass@0 on the same prompts.

pattern also remains similar after removing the two composition-limited models, leaving 914 failures (Appendix G).

These results lead to two observations. First, relative-spatial failure is not a single phenomenon. Different failures arise from different mechanisms, so an intervention targeting one mechanism need not help the others. Second, a substantial fraction of failures depends on the verifier boundary. Of the 339 H_{conv} failures, 316 involve separation relations, where the relaxed criterion allows up to 12 px of overlap. These programs are close to satisfying the strict relation, but are still counted as failures because of the chosen tolerance.

The cause distribution also varies across models ($\chi^2(18) = 248.3$, $p < 0.0001$). For example, H_{ref} accounts for 33.2% of failures in Qwen3-4B but only 2.2% in Qwen2.5-32B-AWQ. In contrast, 76.1% of Qwen2.5-32B-AWQ failures are attributed to H_{extent}. This suggests that stronger models do not simply make fewer spatial errors; the types of errors they make can also change. As a result, two models with similar pass rates may fail for very different reasons. Appendix I breaks the cause mix down by spatial relation.

5.5 E5: An Independent Criterion Agrees

The static provenance axis and the counterfactual one — which never reads the code, only re-executes the program in perturbed scenes — agree on 95.2% of the 1456 attributed failures decidable by both, $\kappa = 0.868$; two carry no animation call on the decisive axis and are therefore not decidable on the static side. The provenance axis is not an artefact of one parser's reach.

6 Analysis

6.1 E6: Aggregate Effects of Prompting Probes

Read the way the field reads interventions — as a change in pass rate — the four probes are discouraging (Table 8 and Figure 7, appendix). Pooled over models and all spatial items, naming the compared edges costs 13.1 points and forcing a grounding header 5.8, while disclosing the verifier's numeric conventions gains 3.1 and handing the model the gold predicate program 3.0 (all $p < 0.0001$). The last is the sharpest single reading: a generator given the exact formal constraint it is graded against writes barely better spatial code than one given the sentence, so having the constraint in formal hand is not what separates a passing program from a failing one here. These probes also act on the form of the prompt and not only on its content — oracle injection costs 17.1 and 17.8 points on the atomic and temporal categories, where the baseline averages 94.9% and 73.5% across the seven models (Table 1) — which is a further reason to read them per cause rather than as one number.

6.2 E7: The Aggregate Is Two Opposite Effects Cancelling

That aggregate is misleading, and the attribution makes it visible. Table 9 (appendix) splits baseline failures by assigned cause and asks what each probe does to exactly those items, alongside what it

does to items the baseline already passed. Every probe repairs a substantial share of the failures it targets, and the ordering by effect differs by cause. The grounding header is the best repair for H_{conv} (33.6%) and H_{ref} (25.6%); convention disclosure is best on H_{extent} (33.5%), which is coherent — stating that "to the left of" means edge-to-edge *is* extent information delivered as a threshold. The explicit rewrite is the worst probe on H_{conv} by a wide margin (14.5% against 33.6%), exactly as the hypothesis predicts: no amount of edge-naming tells a model where a threshold sits.

Every probe also breaks items that already worked: 37.1% of previously passing spatial items under the explicit rewrite, 29.0% under the grounding header, 17.3% under convention disclosure, 13.7% under oracle injection. The small aggregate effects of probes C and O are not inertness but a repair rate and a regression rate of similar size nearly cancelling — the pass rate alone would have called these interventions unimportant, when each does two large and opposite things.

6.3 E8: Why Being Explicit Backfires

For every separation constraint we record the signed slack: the distance from the placement to the boundary the verifier tests, positive when satisfied. Under the baseline the median slack is $+5$ px and 34.8% of constraints are violated; under the explicit rewrite the median is -5 px and 50.5% are violated (Figure 5b, appendix). Naming the compared edges invites the model to place them exactly against each other; a vague instruction invites a comfortable margin. On a criterion with a hard boundary, precision without a tolerance budget is a liability — a property of how the constraint is stated, not of the model's spatial ability.

6.4 E9: What the Program Says That the Trace Cannot

The two axes disagree in a structured way (Table 3, appendix; independence rejected, $\chi^2(2) = 52.1$, $p < 0.0001$), and this is also where the three causes are shown to co-occur rather than partition the phenomenon. Of 681 failures whose placement is extent-ignoring, 537 — 78.9% — are programs that never mention any geometric quantity of the referent: the model did not consult the referent's centre and forget its width, it computed a number that happened to be centre-like. Two programs producing the identical wrong video differ in whether the target position was computed from the referent's geometry, and only one generalises to an unseen scene — invisible downstream of execution, and the part of the diagnosis a pixel-level evaluator cannot make.

Because the two axes disagree this often, the decision order that combines them has a measurable footprint, and the crosstab is what lets a reader recover either reading. Resolving provenance before behaviour — asking first what the program consulted, then what the placement did — relabels 1067 of the 1458 failures (73.2%), almost all of them into H_{ref} (506 from H_{extent}, 228 from H_{conv}, 192 from the residue), and would put H_{ref} at 79.1%. The two orders are not competing estimates of one quantity: behaviour-first names what the placement got wrong, provenance-first names what the program failed to consult, and a program can be both. We report the behaviour-first order because a syntactic criterion alone mislabels the programs that constant-fold the referent's geometry into a literal, and we ship the joint table so the alternative reading is recoverable.

7 Conclusion

Relative-spatial failure in generated animation code is not one thing. REFPROBE names what a failure is, under a decision order fixed before generation and with an error rate measured against injected faults and against an independent counterfactual criterion ($\kappa = 0.868$). Applied to 42,000 programs from seven generators, it splits 1458 relative-spatial failures into a strongly non-uniform distribution — 44.4% extent-ignoring, 23.3% decided at the verifier's boundary, 15.6% unanchored, 16.8% residue — whose shape differs across models and moves with scale. Read differentially, the same split shows the aggregate pass rate is the wrong instrument for evaluating spatial interventions: four probes whose aggregate effect runs from -13.1 to $+3.1$ points each repair a seventh to a third of the failures in a named cause and break a comparable share of what already worked.

AI USE STATEMENT

In this work, we used generative AI tools for language editing, code debugging, and improving the presentation of figures and tables. We also used them for limited brainstorming. All AI-assisted content was reviewed and verified by the authors, including code, experimental results, references, and scientific claims. The authors made all final decisions regarding the methodology, experiments, and conclusions, and take full responsibility for the final content of this work.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program Synthesis with Large Language Models. *arXiv preprint arXiv:2108.07732*, 2021.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code Generation with Generated Tests. *arXiv preprint arXiv:2207.10397*, 2022.

Junhao Chen, Kejun Gao, Yuehan Cui, Mingze Sun, Mingjin Chen, Shaohui Wang, Xiaoxiao Long, Fei Ma, Qi Tian, Ruqi Huang, and Hao Zhao. LottieGPT: Tokenizing Vector Animation for Autoregressive Generation. *arXiv preprint arXiv:2604.11792*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*, 2021.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching Large Language Models to Self-Debug. *arXiv preprint arXiv:2304.05128*, 2023.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. doi: 10.1177/001316446002000104.

Xueying Du, Mingwei Liu, Kaixin Wang, Hanlin Wang, Junwei Liu, Yixuan Chen, Jiayi Feng, Chaofeng Sha, Xin Peng, and Yiling Lou. ClassEval: A Manually-Crafted Benchmark for Evaluating LLMs on Class-level Code Generation. *arXiv preprint arXiv:2308.01861*, 2023.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. DeepSeek-Coder: When the Large Language Model Meets Programming – The Rise of Code Intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, Yaohui Wang, Xinyuan Chen, Ying-Cong Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench++: Comprehensive and Versatile Benchmark Suite for Video Generative Models. *arXiv preprint arXiv:2411.13503*, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder Technical Report. *arXiv preprint arXiv:2409.12186*, 2024.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. *arXiv preprint arXiv:2403.07974*, 2024.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *arXiv preprint arXiv:2310.06770*, 2023.

Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? Investigating their struggle with spatial reasoning. *arXiv preprint arXiv:2310.19785*, 2023.

Max Ku, Thomas Chong, Jonathan Leung, Krish Shah, Alvin Yu, and Wenhu Chen. TheoremExplainAgent: Towards Video-based Multimodal Explanations for LLM Theorem Understanding. *arXiv preprint arXiv:2502.19400*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. *arXiv preprint arXiv:2309.06180*, 2023.

Stephen C. Levinson. *Space in Language and Cognition: Explorations in Cognitive Diversity*. Cambridge University Press, 2003. doi: 10.1017/CBO9780511613609.

Boyu Li, Linjie Qiu, Lin-Ping Yuan, Duotun Wang, Yue Jiang, Zeyu Wang, and Hongbo Fu. Spatula: Exploring On-Demand In-Situ Interfaces and Interaction for Attribute Control. *arXiv preprint arXiv:2607.10405*, 2026a.

Boyu Li, Lin-Ping Yuan, Zeyu Wang, and Hongbo Fu. SketchDynamics: Exploring Free-Form Sketches for Dynamic Intent Expression in Animation Generation. *arXiv preprint arXiv:2601.20622*, 2026b.

Yuejia Li, Ke He, Junheng Li, Shutong Chen, Jingkang Xia, Zhiyue Su, Junchi Zhang, and Mang Ye. See Before You Code: Learning Visual Priors for Spatially Aware Educational Animation Generation. *arXiv preprint arXiv:2605.15585*, 2026c.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

Fangyu Liu, Guy Emerson, and Nigel Collier. Visual Spatial Reasoning. *arXiv preprint arXiv:2205.00363*, 2022.

Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. EvalCrafter: Benchmarking and Evaluating Large Video Generation Models. *arXiv preprint arXiv:2310.11440*, 2023a.

Yuanxin Liu, Lei Li, Shuhuai Ren, Rundong Gao, Shicheng Li, Sishuo Chen, Xu Sun, and Lu Hou. FETV: A Benchmark for Fine-Grained Evaluation of Open-Domain Text-to-Video Generation. *arXiv preprint arXiv:2311.01813*, 2023b.

Jiaju Ma and Maneesh Agrawala. MoVer: Motion Verification for Motion Graphics Animations. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2025.

Jiaju Ma, R. Kenny Jones, Jiajun Wu, and Maneesh Agrawala. Self-Consistency for LLM-Based Motion Trajectory Generation and Verification. *arXiv preprint arXiv:2603.29301*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. *arXiv preprint arXiv:2303.17651*, 2023.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjamshidi. SpartQA: A Textual Question Answering Benchmark for Spatial Reasoning. *arXiv preprint arXiv:2104.05832*, 2021.

Tanawan Premsri and Parisa Kordjamshidi. FoREST: Frame of Reference Evaluation in Spatial Reasoning Tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. doi: 10.18653/v1/2025.emnlp-main.1772.

Qwen Team, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024.

Xinyu Shi, Li-Yi Wei, Nanxuan Zhao, Jian Zhao, and Rubaiat Habib Kazi. Notational Animating: An Interactive Approach to Creating and Editing Animation Keyframes. *Proceedings of the ACM on Human-Computer Interaction*, 2026. doi: 10.1145/3772318.3790707.

Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. StepGame: A New Benchmark for Robust Multi-Hop Spatial Reasoning in Texts. *arXiv preprint arXiv:2204.08292*, 2022.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2303.11366*, 2023.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2Code: Benchmarking Multimodal Code Generation for Automated Front-End Engineering. *arXiv preprint arXiv:2403.03163*, 2024.

Kaiyue Sun, Kaiyi Huang, Xian Liu, Yue Wu, Zihan Xu, Zhenguo Li, and Xihui Liu. T2V-CompBench: A Comprehensive Benchmark for Compositional Text-to-video Generation. *arXiv preprint arXiv:2407.14505*, 2024.

Leonard Talmy. How language structures space. In Herbert L. Pick and Linda P. Acredolo (eds.), *Spatial Orientation: Theory, Research, and Application*, pp. 225–282. Springer, 1983. doi: 10.1007/978-1-4615-9325-6_11.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 Furious. *arXiv preprint arXiv:2501.00656*, 2025.

Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Yixuan Li, and Neel Joshi. Is A Picture Worth A Thousand Words? Delving Into Spatial Reasoning for Vision Language Models. *arXiv preprint arXiv:2406.14852*, 2024.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. doi: 10.1080/01621459.1927.10502953.

Chengyue Wu, Yixiao Ge, Qiushan Guo, Jiahao Wang, Zhixuan Liang, Zeyu Lu, Ying Shan, and Ping Luo. Plot2Code: A Comprehensive Benchmark for Evaluating Multi-modal Large Language Models in Code Generation from Scientific Plots. *arXiv preprint arXiv:2405.07990*, 2024.

Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. SVGDreamer: Text Guided SVG Generation with Diffusion Model. *arXiv preprint arXiv:2312.16476*, 2023.

Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering LLMs to Understand and Generate Complex Vector Graphics. *arXiv preprint arXiv:2412.11102*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Cheng Yang, Chufan Shi, Yaxin Liu, Bo Shui, Junjie Wang, Mohan Jing, Linran Xu, Xinyu Zhu, Siheng Li, Yuxiang Zhang, Gongye Liu, Xiaomei Nie, Deng Cai, and Yujiu Yang. ChartMimic: Evaluating LMM's Cross-Modal Reasoning Capability via Chart-to-Code Generation. *arXiv preprint arXiv:2406.09961*, 2024.

Qiran Zhang, Yuheng Wang, Runde Yang, Lin Wu, Jingru Fan, Shu Yao, Jie Zhang, Tianle Zhou, Huatao Li, Ruijie Shi, Yihan Li, and Chen Qian. PRISM: A Benchmark for Programmatic Spatial-Temporal Reasoning. *arXiv preprint arXiv:2605.19382*, 2026.

Zheng Zhang. Comprehension Without Competence: Architectural Limits of LLMs in Symbolic Computation and Reasoning. *arXiv preprint arXiv:2507.10624*, 2025.

Dian Zheng, Ziqi Huang, Hongbo Liu, Kai Zou, Yinan He, Fan Zhang, Lulu Gu, Yuanhan Zhang, Jingwen He, Wei-Shi Zheng, Yu Qiao, and Ziwei Liu. VBench-2.0: Advancing Video Generation Benchmark Suite for Intrinsic Faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. VGBench: Evaluating Large Language Models on Vector Graphics Understanding and Generation. *arXiv preprint arXiv:2407.10972*, 2024.

A Carrier and Probe Set Construction

A scene holds four labelled objects with distinct colours and shapes, positioned by bounding-box centre on an 800×600 canvas. The runtime exposes four calls: `move`, `resize` and `fade` take absolute targets and `rotate` an angular delta, and the prompt states this. Relations are balanced across four separation relations, three proximity and alignment relations, and four canvas-anchored relations. Every item is generated so that its target relation is false in the initial scene and a satisfying placement exists on canvas; the reference solver confirms this on all 1000 items.

Table 3: Validation of the failure-attribution framework. **(a)** Cross-tabulation between observed placement behaviour and whether the generated program is anchored to the referenced object; independence is rejected ($\chi^2(2) = 52.1$, $p < 10^{-4}$). **(b)** Discriminator validity on programs with known injected causes (row %). Reference programs satisfy the strict verifier on 1000/1000 tasks. Per-class precision/recall is 1.00/0.81 for H_{ref}, 0.88/0.99 for H_{extent}, and 0.97/0.99 for H_{conv}. Static and counterfactual provenance agree on 95.2% of model failures ($\kappa = 0.868$, $n = 1456$).

(a)

Placement behaviour	Anchored	Not anchored	Total
Tolerance-only	112	237	349
Extent-ignoring	144	537	681
Neither	47	381	428

(b)

Injected cause	H_{ref}	H_{extent}	H_{conv}	Uncl.
H_{ref} (absolute target)	81.4	13.4	4.4	0.8
H_{extent} (centre only)	0.0	99.4	0.0	0.6
H_{conv} (inside relaxed)	0.0	0.0	99.3	0.7

B Validity of the Probe Set

Table 3 reports the reference solver against the strict verifier and the discriminator against programs whose cause is known by construction.



Figure 3: Construct validity of the discriminator: rows are programs carrying a known injected cause, columns the label assigned (row %). The single soft cell is injected-H_{ref} read as H_{extent}, the boundary discussed in §5.

C Verifier Conventions

Table 4 lists both frozen settings for every predicate. The strict column is what a benchmark would ship; the relaxed column is the band whose satisfaction defines H_{conv}.

Table 4: The frozen verifier conventions. *Strict* is what a benchmark would ship; *relaxed* is the band whose satisfaction defines H_{conv}. Separation relations have a strict threshold of zero, so their relaxation is an absolute overlap allowance rather than a factor.

Family	Predicate	Strict	Relaxed	Unit
Separation	left_of	0	12	px (abs.)
	right_of	0	12	px (abs.)
	above	0	12	px (abs.)
	below	0	12	px (abs.)
Proximity / alignment	near	20	60	px (3.0×)
	aligned_x	5	25	px (5.0×)
	aligned_y	5	25	px (5.0×)
Canvas-anchored	at_top	60	120	px (2.0×)
	at_bottom	60	120	px (2.0×)
	at_left_border	40	96	px (2.4×)
	at_right_border	40	96	px (2.4×)
Atomic	move_by	5	20	px (4.0×)
	rotate_by	3	12	deg (4.0×)
	fade_to	0.05	0.2	opacity (4.0×)
Temporal	after	0	0.2	s (abs.)

D Models and Serving

All seven generators are served locally with vLLM (Kwon et al., 2023) in bfloat16 with greedy decoding, seed 0, a 2816-token context and a 1024-token generation budget. Table 5 lists each generator with its family, parameter count and serving precision.

E Paired Comprehension–Generation Gap

Table 6 gives the per-model, per-category gap behind §5.

Table 5: Open-weight generators evaluated. All are served locally; decoding is greedy. Par. is the parameter count and Prec. the serving precision.

Model	Family	Par.	Prec.
DS-Coder-6.7B	DeepSeek	6.7B	bf16
Olmo-3-7B	OLMo	7.3B	bf16
Qwen2.5-Coder-7B	Qwen2.5	7.6B	bf16
Qwen2.5-7B	Qwen2.5	7.6B	bf16
Qwen2.5-32B-AWQ	Qwen2.5	32.8B	AWQ4
Qwen3-4B	Qwen3	4.0B	bf16
Qwen3-8B	Qwen3	8.2B	bf16

Table 6: Paired comprehension–generation gap on the *same* prompts: predicate-synthesis accuracy (P, gold predicate set recovered) minus code pass@0 (C), in points. Positive means the model states the constraint it cannot write.

Model		Atomic	Temporal	Spatial	Sp.-temp.	Compound	All
DeepSeek-Coder-6.7B-Instruct	P	100.0	100.0	89.5	55.5	99.5	88.9
	C	75.5	38.0	64.5	27.5	37.0	48.5
	Δ	+24.5	+62.0	+25.0	+28.0	+62.5	+40.4
Olmo-3-7B-Instruct	P	98.5	94.0	78.0	71.5	99.5	88.3
	C	95.5	1.0	74.5	2.5	21.5	39.0
	Δ	+3.0	+93.0	+3.5	+69.0	+78.0	+49.3
Qwen2.5-Coder-7B-Instruct	P	100.0	100.0	80.5	10.5	85.5	75.3
	C	99.5	94.5	80.5	68.5	53.5	79.3
	Δ	+0.5	+5.5	+0.0	-58.0	+32.0	-4.0
Qwen2.5-7B-Instruct	P	100.0	99.5	84.5	70.0	70.0	84.8
	C	96.0	90.0	73.5	58.0	24.0	68.3
	Δ	+4.0	+9.5	+11.0	+12.0	+46.0	+16.5
Qwen2.5-32B-Instruct-AWQ	P	100.0	100.0	79.0	51.0	99.0	85.8
	C	100.0	100.0	95.5	85.5	72.5	90.7
	Δ	+0.0	+0.0	-16.5	-34.5	+26.5	-4.9
Qwen3-4B	P	100.0	99.5	100.0	16.5	74.5	78.1
	C	98.0	92.0	90.0	69.5	41.0	78.1
	Δ	+2.0	+7.5	+10.0	-53.0	+33.5	+0.0
Qwen3-8B	P	100.0	100.0	100.0	77.0	97.5	94.9
	C	99.5	99.0	89.0	78.0	44.5	82.0
	Δ	+0.5	+1.0	+11.0	-1.0	+53.0	+12.9

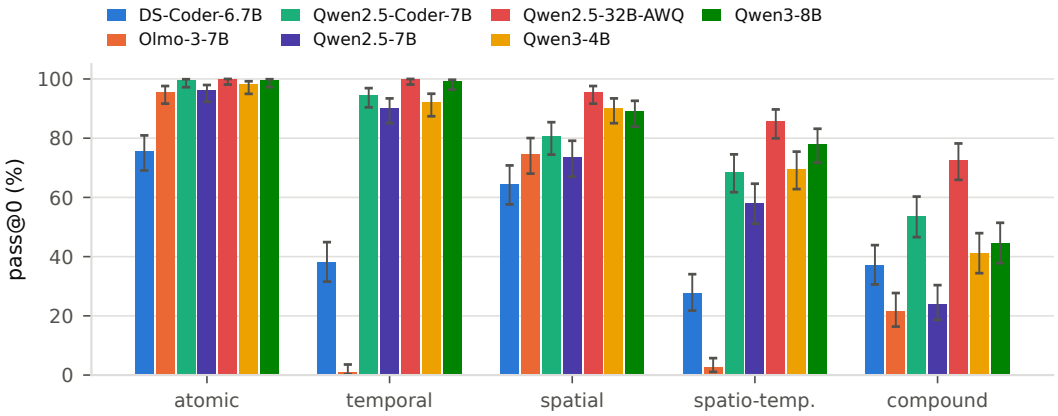


Figure 4: First-attempt verification rate by instruction category, with 95% Wilson intervals.

F Comprehension-Side Premature Grounding

Table 7a gives the per-model rate at which a relational instruction is translated into an absolute displacement.

G Cause Mix Under Each Condition

Table 7b gives the mix under every instruction condition. Restricting the pooled attribution to the five Qwen models — dropping the two whose binding constraint is composition rather than space — leaves 914 failures distributed 51.2% H_{extent}, 24.5% H_{conv}, 14.3% H_{ref} and 10.0% residue, the same ordering as the pooled result.

Table 7: Analysis of spatial-relation failures. **(a)** Premature grounding on the comprehension side: when asked only to translate a relative-spatial instruction into a predicate, models may replace the relation with an invented absolute displacement. Counts are over items whose gold program contains a relational spatial predicate. **(b)** Distribution of attributed failure causes under different probe conditions, pooled over models; values report the share of failures (%) assigned to each hypothesis.

(a)

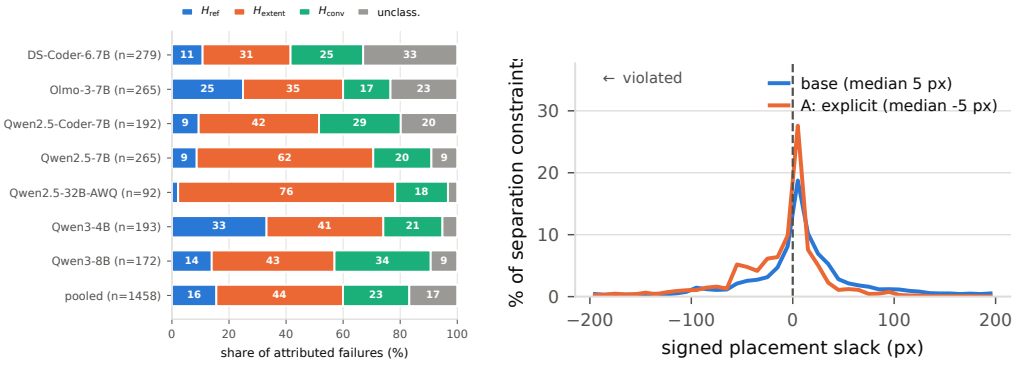
Model	n	kept	grounded
DS-Coder-6.7B	501	89.6	5.6
Olmo-3-7B	501	98.8	0.0
Qwen2.5-32B-AWQ	501	89.2	10.8
Qwen2.5-7B	501	83.8	15.4
Qwen2.5-Coder-7B	501	67.3	31.9
Qwen3-4B	501	65.5	33.3
Qwen3-8B	501	94.4	5.2
Pooled	3507	84.1	14.6 [13.5,15.8]

(b)

Condition	n	H_{ref}	H_{extent}	H_{conv}	Uncl.
baseline	1458	15.6	44.4	23.3	16.8
A: explicit	1729	10.4	53.1	19.6	16.9
B: header	1535	11.6	51.7	17.5	19.3
C: convention	1409	14.4	44.5	30.5	10.6
O: oracle	1369	15.7	52.0	23.1	9.2

H Probe Effects and Repair by Cause

Table 8 presents the pooled change in first-attempt pass rate that each probe causes, per instruction category, against the baseline instruction on the same prompts. Table 9 splits the baseline failures by assigned cause and gives the share of each cause that a probe repairs, together with the share of already-passing spatial items that it breaks.



(a) Cause attribution across models and pooled.

(b) Signed slack of separation constraints.

Figure 5: Analysis of relative-spatial failures. **(a)** Cause attribution varies substantially across models, while the pooled distribution is strongly non-uniform. **(b)** Signed slack measures the distance from the verifier's separation boundary; explicitly naming the compared edges shifts the distribution toward and beyond the boundary.

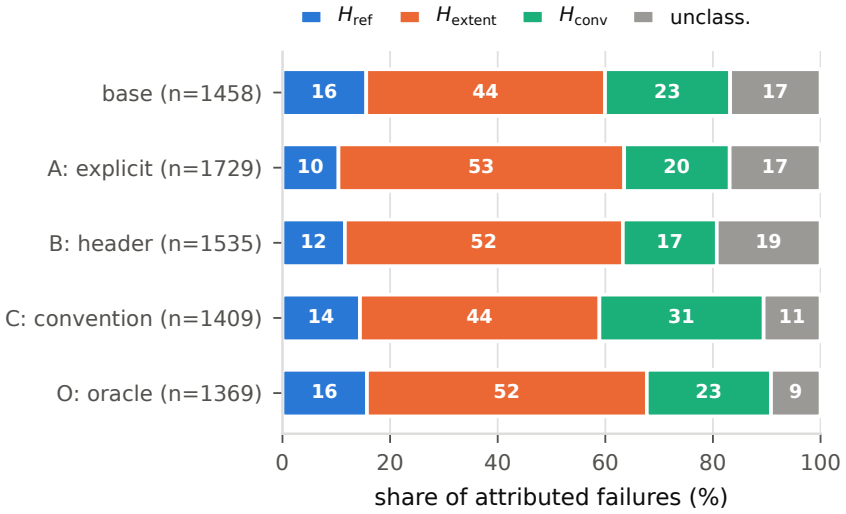


Figure 6: Cause mix under each instruction condition, pooled over models.

Table 8: Probe effects, pooled over models: change in pass@0 against the baseline instruction on the *same* prompts (points), with the exact McNemar p. A probe that supplies information on the input side (A) is not equivalent to one that forces it on the output side (B).

Condition	Atomic	Temporal	Spatial	Sp.-temp.	Compound	All spatial
A: explicit rewrite	+0.4 ($p = 0.6398$)	-8.0 ($p < 0.0001$)	-12.7 ($p < 0.0001$)	-12.4 ($p < 0.0001$)	-14.3 ($p < 0.0001$)	-13.1 ($p < 0.0001$)
B: grounding header	-2.2 ($p = 0.0164$)	-8.1 ($p < 0.0001$)	-5.9 ($p < 0.0001$)	-4.5 ($p = 0.0006$)	-7.1 ($p < 0.0001$)	-5.8 ($p < 0.0001$)
C: convention disclosed	-1.9 ($p = 0.0019$)	+5.7 ($p < 0.0001$)	-2.1 ($p = 0.0652$)	+9.9 ($p < 0.0001$)	+1.4 ($p = 0.3493$)	+3.1 ($p < 0.0001$)
O: oracle predicate	-17.1 ($p < 0.0001$)	-17.8 ($p < 0.0001$)	+1.5 ($p = 0.1259$)	+9.4 ($p < 0.0001$)	-1.9 ($p = 0.1553$)	+3.0 ($p < 0.0001$)

I WHERE THE CAUSES LIVE

Attribution by relation (Figure 8) is coherent with the mechanism. Separation relations, whose satisfaction depends on both extents, concentrate H_{extent}: 206 of 314 *left-of* and 223 of 447 *above* failures. *Near* produces none — an extent-ignoring placement of "near" overlaps the referent and therefore passes — and yields H_{ref} instead (59 of 89). Alignment relations are extent-independent by construction, and canvas-anchored relations rarely fail at all (34 attributed failures across seven

Table 9: What each probe does once failures are split by cause. Left: of the baseline failures the discriminator assigned to a cause, the share that pass under the probe. Right: of the spatial items the baseline already passed, the share the probe breaks. Every probe both repairs previously failing items and causes regressions on previously passing items.

	Repaired				
Probe	H_{ref}	H_{extent}	H_{conv}	Uncl.	Broken
A: explicit rewrite	22.9	23.8	14.5	35.9	37.1
B: grounding header	25.6	31.7	33.6	25.7	29.0
C: convention disclosed	19.8	33.5	24.8	38.4	17.3
O: oracle predicate	23.3	19.6	23.3	40.8	13.7
n	227	647	339	245	2,502

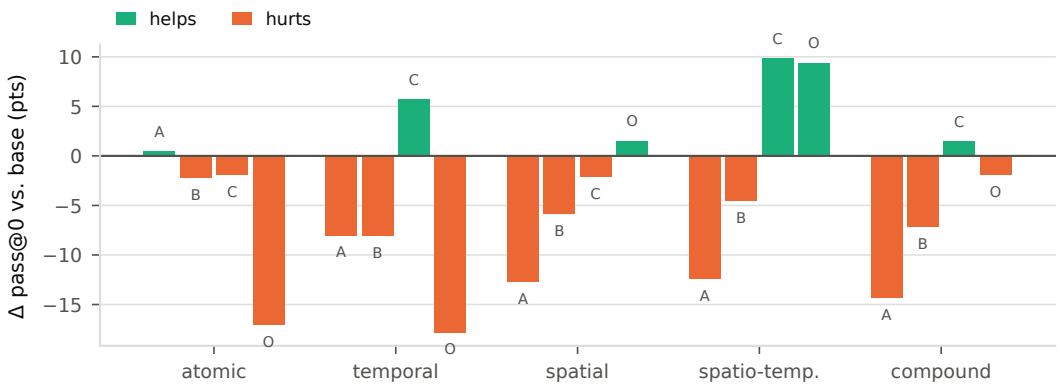


Figure 7: Change in pass@0 against the baseline instruction on the same prompts, pooled over models. Read in aggregate, every probe is inert or harmful.

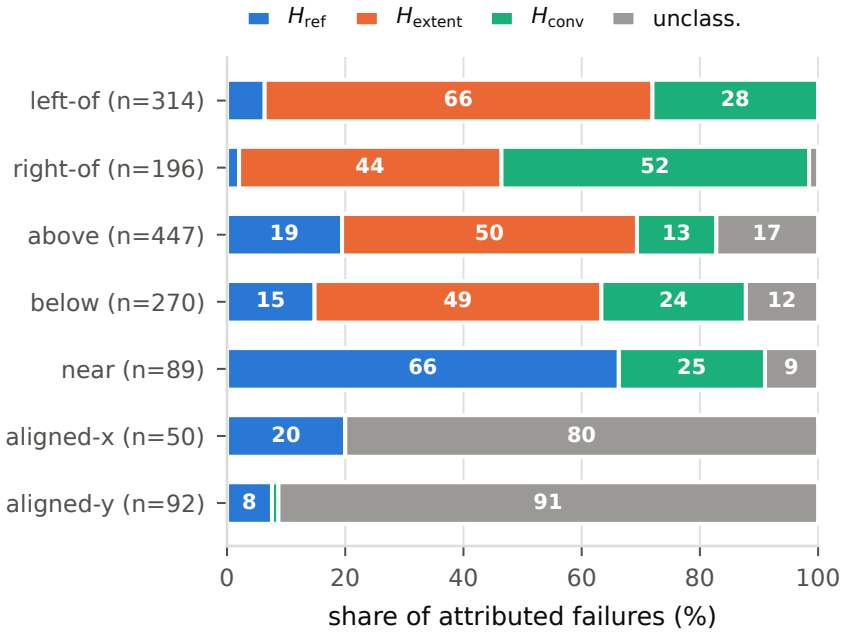


Figure 8: Cause of failure by spatial relation. The mix follows what each relation makes decidable: separation relations concentrate H_{extent}, proximity relations cannot produce it, and alignment relations are extent-independent by construction.

models), because an object placed loosely near an edge still satisfies an edge predicate. The cause mix is a property of what the relation makes decidable — a further reason not to report one spatial pass rate. The probes also change the mix, not just the level (Table 7b, Figure 6): under convention disclosure the H_{conv} share rises from 23.3% to 30.5% and the residue falls from 16.8% to 10.6%.