STAGE: Diagnosing Semantic Transfer at Grounded Execution in Embodied Agents

Baosheng Jin **Yushen Liang** **Hua Shen**
Center for Data Science, NYU Shanghai
Shanghai, China
{bj2410, yl11949, hs3645}@nyu.edu

Abstract

Embodied language grounding requires more than identifying the referent of an instruction: recovered semantics must also control the action an agent exposes. We study this missing link as a **semantic–action gap**, where instruction semantics are recoverable but weakly expressed in native continuous actions. We introduce **SAT-Bench**, a fixed-observation counterfactual benchmark that holds the visual scene and agent state fixed while changing only instruction semantics. On LIBERO target-name and pixel-grounded relation swaps, target recovery reaches 100.0% and 95.8%, whereas OpenVLA action sensitivity remains only 6.8% and 7.7%. The gap persists across 1,000 additional compositional and temporal/procedural counterfactuals, with overall action sensitivity of 6.1%. Hidden-state, threshold-free, cross-policy, and rollout diagnostics further support this semantic–action transfer failure. We introduce **VISA**, a lightweight execution-time interface that converts recovered semantics into ALLOW, DEFER, target-consistency, and verified-selection decisions. VISA reduces invalid-instruction blind execution from 92.7% to 2.8% while preserving 94.0% of normal commands, and verified selection further improves target-consistent action exposure without updating the underlying policy. Overall, embodied language evaluation should measure semantic–action transfer, not semantic parsing alone.

1 Introduction

Multimodal language-grounding evaluations often ask whether a model identifies the correct object, relation, or instruction state (Goyal et al., 2017; Agrawal et al., 2018; Thrush et al., 2022; Su et al., 2024). Embodied language agents add an action-interface requirement: recovered semantics should affect whether and how an action is exposed. However, a vision-language-action (VLA) policy (Kim et al., 2025; Ghosh et al., 2024) may recover the requested target and still move toward a plausible object from scene priors, or may parse a prohibited command and still expose an executable motion. We therefore ask an action-boundary question: *when the world is fixed, does changing instruction semantics change the native continuous action accordingly?*

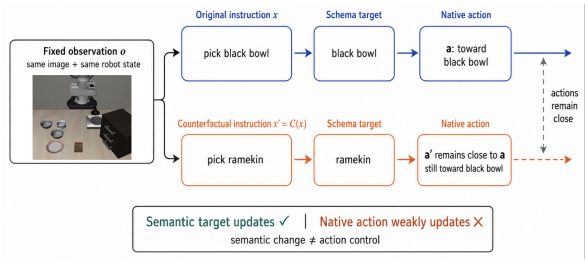

Figure 1: Fixed-observation counterfactual protocol. The observation and agent state are fixed while only the instruction semantics changes.

We call this interface-level requirement *semantic–action transfer* (SAT): recovered instruction semantics should affect whether and how a native action is exposed. When semantics are recoverable but remain weakly expressed in the policy's native seven-degree-of-freedom (7-DoF) action, we say the policy exhibits a *semantic–action gap* (SAG). To measure this gap, we use a *fixed-observation counterfactual protocol*: holding the observation and agent state fixed, we change only instruction semantics and test whether the exposed native action changes accordingly. Figure 1 illustrates this protocol, which separates semantic recoverability from semantic control of action. This distinction is the central thesis of the paper: semantic availability is not semantic action control, so embodied language evaluation should measure semantic-to-action transfer rather than semantic parsing or task success alone.

This paper makes this action-boundary problem measurable in two steps: first through a benchmark for diagnosing semantic–action transfer, and then through an execution-time interface that regulates action exposure. We introduce SAT-

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 24309–24331
October 24-29, 2026 ©2026 Association for Computational Linguistics

BENCH (*Semantic–Action Transfer Benchmark*), a fixed-observation counterfactual benchmark spanning referential, relational, compositional, temporal/procedural, invalid, prohibitive, and paraphrase-preserving instruction changes. On LIBERO (Liu et al., 2023) target-name and pixel-grounded relation swaps, Qwen2.5-VL (Bai et al., 2025) recovers the intended target in 100.0% and 95.8% of cases, yet OpenVLA action sensitivity is only 6.8% and 7.7%, respectively. The gap persists across 1,000 additional compositional and temporal/procedural counterfactuals, with overall action sensitivity of 6.1%. Hidden-state probes, threshold-free analyses, schema–action attribution, and short-horizon rollouts test alternative explanations based on absent semantic decodability, threshold choice, grounding failure, and one-step artifacts. Importantly, probing is used only as evidence that semantic information is decodable from model representations, not that the action decoder causally relies on it. The same action-boundary failure also appears for non-executable language, where invalid or prohibited instructions can still trigger blind native execution.

These diagnostics motivate an explicit semantic–action interface: recovered semantics should not only be readable, but should regulate which actions are exposed. We therefore introduce VISA (*Verifiable Interface for Semantic Action*), a lightweight execution-time interface that extracts a compact instruction-conditioned schema, checks admissibility, passes through valid native actions, defers invalid ones, and flags target-inconsistent actions. A verified-selection extension further uses the recovered semantics to select among candidate actions before exposure. VISA does not update the underlying VLA policy or claim to solve low-level or long-horizon control; its intervention target is the action boundary itself. Empirically, VISA reduces invalid-instruction blind execution from 92.7% to 2.8% while preserving 94.0% of normal commands, and verified selection further improves target-consistent action exposure. Thus, SAT-BENCH diagnoses semantic–action transfer, while VISA operationalizes the same distinction at execution time.

Our contributions are four-fold:

- a formulation of *semantic–action transfer* and *semantic–action gaps* as interface-level requirements for embodied language grounding;
- SAT-BENCH, a release-ready fixed-observation counterfactual benchmark spanning referential, relational, compositional, temporal/procedural, invalid, prohibitive, and paraphrase-preserving semantics;
- diagnostic evidence that instruction semantics can be recoverable externally and linearly decodable from OpenVLA representations while remaining weakly expressed in native actions, together with behavioral tests of alternative explanations;
- VISA, a schema-based execution-time semantic–action interface for allow, defer, target-consistency, and verified-selection decisions without updating or replacing the underlying low-level policy.

2 Related Work

Grounded language evaluation, counterfactuals, and probing. Multimodal grounding work tests whether models recover intended objects, relations, or scene states from language and perception. Counterfactual and contrastive evaluations further ask whether predictions rely on intended semantic variables rather than dataset priors or superficial correlations (Goyal et al., 2017; Agrawal et al., 2018; Thrush et al., 2022). Related embodied and procedural benchmarks study planning, grounding, scene consistency, and robustness under semantic perturbations (Su et al., 2024; Chakraborty et al., 2025). We target a narrower action-interface question: holding the observation and agent state fixed, does changing only the instruction induce the corresponding change in the exposed action? We also draw on representation probing, which tests whether task-relevant information is present in intermediate states (Alain and Bengio, 2016; Hewitt and Manning, 2019; Belinkov, 2022); in our setting, probing helps separate absent semantic recovery from failed transfer into action (Zhang et al., 2025).

Embodied language agents and VLA benchmarks. Recent embodied foundation models connect visual observations, language instructions, and actions through imitation learning, multimodal pretraining, and large robot datasets (Brohan et al., 2023; Driess et al., 2023; Zitkovich et al., 2023; Open X-Embodiment Collaboration, 2024). OpenVLA and Octo provide open-source VLA policies that enable controlled inspection of language-conditioned action behavior (Kim et al., 2025; Ghosh et al., 2024), while LIBERO offers language-conditioned manipulation tasks for transfer evaluation (Liu et al., 2023). These works primarily evaluate task success, generalization, or imitation

quality. Our focus is complementary: we use native actions as a readout of grounded-language transfer rather than as the sole target of control optimization.

Invalid instructions, structured semantics, and action interfaces. A related line of work studies whether embodied agents can reject false-premise commands, infeasible requests, unsafe instructions, or scene-task inconsistencies (Hsieh et al., 2025; Chakraborty et al., 2025; Son et al., 2025). High-level language-to-robot systems combine language reasoning with affordances, feedback, or programmatic primitives to improve executable planning (Ichter et al., 2023; Huang et al., 2023; Liang et al., 2023). Robotic language understanding has long used structured semantic representations to connect commands with objects, relations, actions, and constraints (Tellex et al., 2011; Thomason et al., 2016, 2019); modern VLMs such as Qwen2.5-VL provide image-conditioned parsing and grounding (Bai et al., 2025). VISA builds on this interface view, but uses schemas for evaluation and exposure decisions: whether recovered semantics should allow, defer, or flag an action.

3 Semantic–Action Transfer Framework

We formalize embodied instruction following as an action-interface problem with two separable requirements: the relevant semantic state must be recoverable, and the exposed continuous action must appropriately reflect that state. The goal is not to optimize robot motion, but to evaluate whether recovered semantics reach the native action interface. We first define semantic–action transfer under fixed observations, then introduce SAT-BENCH interventions, and finally define measurements for action sensitivity, non-execution, and target consistency.

3.1 Interface Requirement under Fixed Observations

Let o denote the visual observation together with the robot state, x a natural-language instruction, and $a = f(o, x)$ the native continuous action predicted by a VLA policy. We study an interface-level requirement: if instruction semantics change while the world state is fixed, the exposed action should respond in the corresponding semantic direction, either by redirecting motion or by inhibiting action exposure when execution is inadmissible.

A counterfactual instruction $x' = C(x)$ changes the requested semantics while keeping o fixed, yielding $a' = f(o, x')$. The fixed-observation design is a measurement protocol rather than the definition of the failure itself. If both the observation and instruction change, an action difference could be caused by language, visual features, object layout, or scene priors. By holding o fixed, the protocol isolates the instruction-conditioned component of action selection.

This formulation separates two questions often conflated in embodied evaluation: whether the relevant semantics are recoverable from (o, x), and whether the native action boundary reflects those semantics. We call the desired interface property *semantic–action transfer* (SAT). When semantic recovery is high but corresponding action control is weak, the policy exhibits a *semantic–action gap* (SAG). Thus, SAG is not a claim that the model lacks language understanding, nor that a particular internal representation is causally ignored; it is a behavioral claim that recovered semantics are weakly expressed at the native action boundary.

3.2 SAT-BENCH Interventions

SAT-BENCH instantiates the fixed-observation protocol through controlled instruction interventions while the observation and robot state remain fixed. We define the intervention family

$$\mathcal{C} = \{C_{\text{name}}, C_{\text{rel}}, C_{\text{comp}}, C_{\text{proc}}, C_{\text{invalid}}, C_{\text{neg}}, C_{\text{para}}\}. \quad (1)$$

with $C \in \mathcal{C}$ probing a distinct semantic-to-action property.

C_{name} replaces the explicitly named target while the scene is fixed, e.g., "pick up the black bowl" becomes "pick up the ramekin," testing whether target identity redirects the exposed action. C_{rel} removes the target name and specifies the target through a spatial relation to an anchor object, testing whether grounded relational semantics reach the same action boundary. C_{comp} changes compositional target semantics through attribute–relation combinations or multiple simultaneous constraints, testing whether richer conjunctions alter action exposure. C_{proc} changes temporal or procedural semantics, including the required first subgoal, action ordering, or procedural constraints, testing whether the exposed action reflects the currently required step. C_{invalid} converts executable commands into blank, impossible, ambiguous, or unsupported instructions, testing whether non-executable semantics inhibit action exposure. C_{neg} constructs prohibited

commands such as `do_not`, `avoid`, `leave_alone`, and `except`, testing whether language can block an otherwise plausible action. Finally, C_{para} produces semantically equivalent paraphrases and serves as a control for wording-induced action variation.

Together, these interventions test three interface behaviors required for semantic–action transfer: redirecting actions when executable semantics change, inhibiting actions when execution becomes inadmissible, and calibrating the amount of native action variation induced by wording alone.

3.3 Measurement Stack

We use a three-part measurement stack. **Action-sensitivity metrics** test whether an executable semantic intervention changes the native action beyond wording-induced variation. **Non-execution metrics** test whether non-executable semantics suppress action exposure. **Target-consistency metrics** test whether an exposed action is directed toward the recovered target rather than a competing target. Together, these measurements separate three interface questions: whether actions respond to semantic changes, whether inadmissible actions are withheld, and whether exposed actions are semantically directed.

Action-sensitivity metrics. For executable semantic interventions, we measure native action sensitivity using a normalized action difference:

$$D(a, a') = \left\| (a - a') \oslash \sigma_a \right\|_2 , \tag{2}$$

where σ_a is the per-dimension action standard deviation for the evaluated policy. An action is counted as sensitive if

$$D(a, a') > \tau_s, \tag{3}$$

where τ_s is calibrated separately for each policy and evaluation setting as the 95th percentile of action deltas from semantics-preserving C_{para} controls. These controls keep the observation and task semantics fixed while varying only wording. Target-changing counterfactuals are never used to set τ_s; calibration and evaluation therefore share the observation distribution but are disjoint in perturbation type. We report the exact setting-specific thresholds in the appendix.

Given a task-appropriate semantic recovery signal $\mathrm{SemSens}$, we quantify the semantic–action gap as

$$\mathrm{SAG} = \mathrm{SemSens} - \mathrm{ActionSens}. \tag{4}$$

For target-name swaps, $\mathrm{SemSens}$ is schema sensitivity; for oracle relation swaps, $\mathrm{SemSens} = 1$; and for pixel-grounded relation swaps, $\mathrm{SemSens}$ is grounding accuracy over all examples or 1 on the grounding-correct subset. For richer compositional and procedural interventions, $\mathrm{SemSens}$ is defined by recovery of the intervention-specific target or required subgoal. To avoid making the diagnosis depend on a single calibrated cutoff, we additionally report target-versus-control AUC over the full action-delta distribution.

Non-execution metrics. For inadmissible instructions, the desired interface behavior is non-execution rather than a different target-directed motion. Let $a_{\text{orig}} = f(o, x)$ be the action under the original valid instruction and $a_{\text{inv}} = f(o, x_{\text{invalid}})$ the action under an invalid perturbation. Using the matched policy- and setting-specific threshold τ_s, we define

$$\begin{aligned} \mathrm{BlindExec} &= \mathbb{I}[D(a_{\text{orig}}, a_{\text{inv}}) < \tau_s] , \\ \mathrm{ActionInhib} &= 1 - \mathrm{BlindExec}. \end{aligned} \tag{5}$$

Blind execution measures whether an invalid instruction still exposes a native action close to the original executable command. Action inhibition measures whether non-executable semantics suppress that exposure. At the schema level, an invalid instruction is rejected if the execution decision is a safe deferral such as `ask`, `hold`, `abort`, or `target-not-found`. We report invalid safe deferral together with normal-command pass rate so that a trivial all-stop system cannot score well by refusing every instruction.

Target-consistency metrics. Action sensitivity only asks whether an action changes; it does not determine whether the changed action is directed toward the intended target. We therefore define ActCheck, a target-consistency diagnostic. Let p_e be the end-effector position, p_t the intended target position, p_w a competing wrong or original target, and $v = a_{1:3}$ the translational action. We compute

$$c_t = \cos(v, p_t - p_e), \qquad c_w = \cos(v, p_w - p_e). \tag{6}$$

With margin δ_{ac}, the action is target-aligned if $c_t - c_w > \delta_{\text{ac}}$, wrong-target if $c_w - c_t > \delta_{\text{ac}}$, and ambiguous otherwise.

Crossing this label with schema correctness yields *schema–action attribution*, which separates semantic-recovery failures from semantic-to-action transfer failures. A wrong schema with a wrong

Policy	Domain	Intervention	N	Sem.	Act.	Gap	ActCheck
OpenVLA base	LIBERO	target-name	600	100.0	6.8	93.2	–
OpenVLA-LIBERO90	LIBERO	target-name	600	100.0	5.7	94.3	–
Octo-small-1.5	LIBERO	target-name	600	100.0	42.7	57.3	–
OpenVLA base	LIBERO	pixel relation	600	95.8	7.7	88.1	45.5
Octo-small-1.5	LIBERO	pixel relation	600	95.8	37.5	58.3	42.8
Octo-small-1.5	BridgeData V2	target/task	165	100.0	66.1	33.9	limited

Table 1: Semantic-to-action transfer across policies and domains. Sem. denotes semantic recovery, Act. denotes native action sensitivity, and Gap is Sem. minus Act.; numeric values except N are percentages.

action is primarily a recoverability failure; a correct schema with a wrong or ambiguous action is an interface-transfer failure; and a correct schema with an aligned action is the desired case. ActCheck is therefore an action-boundary diagnostic rather than a measure of full manipulation success.

3.4 From Diagnostics to VISA

These measurements localize failures at the action boundary. If semantics are unrecoverable, the bottleneck lies in perception or language grounding; if semantics are recoverable but executable semantic changes are weakly reflected in native actions, the bottleneck is semantic-to-action transfer; if inadmissible instructions still expose actions, the bottleneck is admissibility; and if exposed actions point toward the wrong target, the bottleneck is target consistency. These failure modes motivate VISA: an execution-time interface that converts recovered semantics into explicit action-exposure decisions—allowing admissible native actions, deferring inadmissible instructions, flagging target-inconsistent actions, or selecting among verified action proposals—without updating or replacing the underlying low-level policy.

4 Diagnostic Evidence for the Semantic–Action Gap

We ask three questions: whether recoverable semantics control native actions, whether simpler explanations account for the observed gap, and whether the same boundary extends beyond simple target changes. Unless otherwise specified, the action policy is OpenVLA and the schema or grounding probe is Qwen2.5-VL-7B-Instruct. We additionally evaluate OpenVLA-LIBERO90 and Octo-small-1.5, and use BridgeData V2 as a real-robot instruction-change diagnostic rather than a strict object-grounded target-swap benchmark. External probes measure semantic recoverability only; hidden-state probes measure linear decodability rather than causal use by the action decoder. The full diagnostic pipeline is provided in Appendix B.

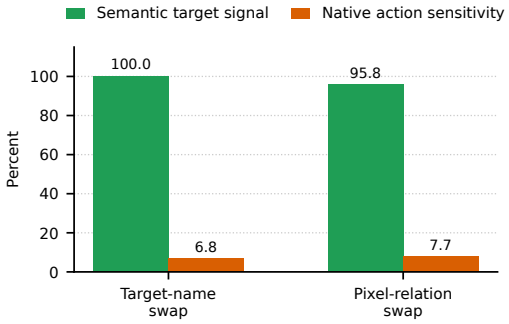

Figure 2: Main semantic–action gap on OpenVLA counterfactuals: semantic target signals remain high while native action sensitivity is low.

4.1 Recoverable Semantics Weakly Control Native Actions

On target-name swaps, the schema target changes in 100.0% of cases while OpenVLA action sensitivity is only 6.8%. On pixel-grounded relation swaps, Qwen2.5-VL recovers the intended target from the image and relation in 95.8% of cases, yet OpenVLA action sensitivity remains 7.7%. The latter setting is especially diagnostic because the target name is absent from the instruction. Thus, high semantic recovery can coexist with weak expression of the same change at the native action boundary.

Table 1 shows that the phenomenon is graded rather than specific to one OpenVLA checkpoint. OpenVLA-LIBERO90 exhibits a similar gap, while Octo-small-1.5 is substantially more action-sensitive. BridgeData V2 shows still greater responsiveness, but should be interpreted only as a real-robot instruction-change diagnostic because it lacks the object metadata required for strict target swaps.

4.2 Ruling Out Alternative Explanations

A low action-sensitivity score admits several alternative explanations. Table 2 summarizes the diagnostic used to test each one.

Schema/action case	Count	Rate (%)
Correct target + aligned action	311	51.8
Correct target + wrong/ambig. action	**264**	**44.0**
Wrong target	25	4.2

Table 3: Schema–action attribution on 600 pixel-grounded relation examples.

Alternative explanation	Diagnostic
Semantics absent from VLA representations	Hidden-state probe
Threshold artifact	Threshold-free AUC
Grounding-only failure	Schema–action attribution
Overlapping target directions	Spatial subsets
First-step irrelevance	Short-horizon rollouts
Simple-semantics only	Richer semantic splits

Table 2: Alternative explanations and the diagnostics used to test them.

Internal decodability. A natural alternative is that only the external Qwen probe recovers the target while OpenVLA itself lacks the corresponding information. Linear probes show otherwise: in the full pixel-relation condition, target schema sensitivity is 1.000 while decoded action sensitivity is 0.094; on an image-dependent subset where language-only or shuffled-image controls fail, the corresponding values are 1.000 and 0.060. Full layer, masking, shuffled-image, and random-label controls are reported in the appendix.

These results establish linear decodability, not causal reliance by the action decoder. Preliminary activation patching likewise has limited effect (Appendix G.5), so we do not claim to identify the complete internal causal mechanism.

Metric and grounding robustness. Threshold-free target-versus-paraphrase comparisons yield AUCs of 57.3 and 58.3 for OpenVLA target-name and pixel-relation swaps, respectively. In contrast, BridgeData V2 with Octo reaches 92.9 (Appendix L.1), showing that the metric can register stronger instruction-conditioned action separation when present. Spatially separated target subsets give the same qualitative OpenVLA result (Appendix D.5).

Schema–action attribution further separates grounding from transfer failures. On the 600 pixel-grounded relation examples, 44.0% have the correct recovered target but a wrong or ambiguous native action, whereas only 4.2% fall in schema-wrong regions.

Short-horizon behavioral relevance. One-step sensitivity is not itself a task-success metric. Across 300 paired OpenVLA-LIBERO90 rollouts of $K = 20$ steps, raw counterfactual instructions do not reliably redirect trajectories. However, first-step ActCheck labels significantly stratify later behavior: target-aligned first actions yield higher counterfactual-target approach and target preference than ambiguous or wrong-target actions. Full rollout statistics are reported in Appendix G.8. We therefore use ActCheck as a short-horizon predictive diagnostic rather than as evidence of full task success.

Split	N	Act. sens. (%)	SAG (%)
Compositional	600	7.0	93.0
Temporal/procedural	400	4.8	95.3
Overall	**1,000**	**6.1**	**93.9**

Table 4: Semantic–action transfer on richer SAT-BENCH interventions.

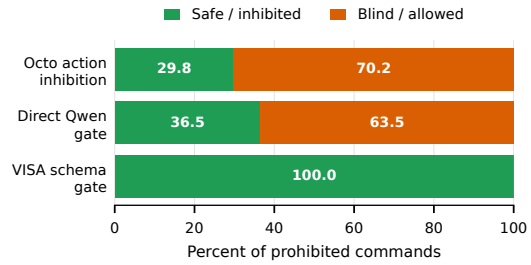

Figure 3: Non-executable language stress test on prohibited commands.

4.3 Beyond Simple Target Changes

We next test semantic breadth along two complementary axes. First, we expand SAT-BENCH with 1,000 fixed-observation counterfactuals covering compositional and temporal/procedural semantics.

The gap remains large: action sensitivity is 7.0% for compositional changes and 4.8% for temporal/procedural changes. The phenomenon therefore extends beyond explicit object replacement and simple spatial relations to conjunctions of semantic constraints and changes in the required first subgoal or action ordering.

Second, the same boundary appears when semantics should inhibit rather than redirect action. OpenVLA blindly executes 92.7% of templated invalid instructions. On 800 Octo negation/prohibition variants, only 29.8% are inhibited, leaving 70.2% blind execution.

Because benchmark-label errors could themselves create an apparent semantic–action gap, we additionally perform blind external validation with three non-author annotators on 300 randomized items, yielding 900 annotation slots. Table 5 summarizes agreement with the benchmark labels.

Agreement is high for executability, intended target, and robot decision. First-subgoal agree-

Field	Agreement	Fleiss' κ	Maj. acc.
Executability	0.964	0.825	1.000
Intended target	0.987	0.975	0.997
First required subgoal	**0.691**	**0.524**	**0.770**
Robot decision	0.978	0.916	1.000

Table 5: Blind external validation with three non-author annotators. The lower first-subgoal agreement reflects ambiguity in procedural decomposition.

ment is lower, reflecting the greater ambiguity of procedural decomposition; we therefore treat temporal/procedural results as controlled diagnostics rather than exhaustive procedural-language evaluation. Full agreement statistics and audit details are reported in Appendix C.6.

Together, these results localize a broader action-boundary failure: semantics can be recoverable through external grounding or linearly decodable from VLA representations, yet remain weakly reflected when action should be redirected, inhibited, or made target-consistent. This diagnosis motivates the execution-time interface introduced next.

5 VISA: A Semantic–Action Interface for Action Exposure

Section 4 localizes semantic–action failures at the action boundary: recovered semantics may fail to redirect native actions, inadmissible instructions may still expose executable motions, and target-correct schemas may still yield wrong-target or ambiguous actions. VISA responds with an explicit execution-time interface that makes recovered semantics inspectable and uses them to regulate action exposure. It passes through admissible native actions, defers inadmissible instructions, and flags target-inconsistent actions before execution. Figure 4 summarizes the interface.

5.1 Schema, Admissibility, and Action Exposure

Schema extraction. For a task family τ, VISA extracts an instruction-conditioned schema

$$s_\tau = (t, e, A^+, A^-, u, d, r), \tag{7}$$

where t denotes the target, e target existence, A^+ and A^- allowed and blocked symbolic actions, u the next symbolic action, d the execution decision, and r a reason. Task families expose only the fields required by their semantics: relation tasks additionally represent anchor and relation variables, while temporal/procedural tasks represent the required subgoal or ordering constraint.

Schemas are specified once per task family rather than annotated per instance. Table 6 summarizes

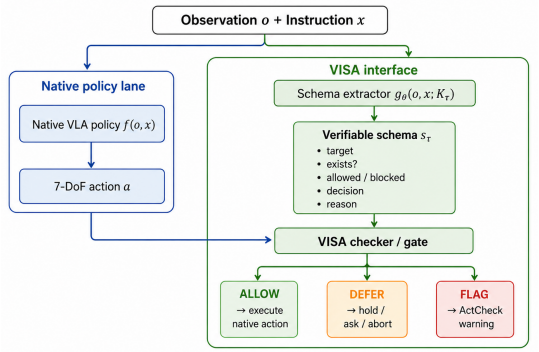

Figure 4: VISA semantic–action interface: schema extraction, admissibility checking, and action exposure.

the corresponding design effort. Across the families studied here, schema construction requires low-to-medium manual effort and no instance-level schema labels.

Task family	Additional semantic fields	Effort	Inst. labels
Target-name	target / existence	low	no
Relation	anchor / relation	low–med.	no
Invalid / prohibition	blocked actions / decision	low	no
Compositional	attributes / constraints	low–med.	no
Temporal / procedural	subgoal / ordering	medium	no

Table 6: Task-family schema design requirements. Schemas are specified once per family rather than annotated per instance.

Do the schema fields matter? We evaluate this question with a controlled 300-example target-field ablation. Table 7 reports the change in candidate recall between the complete schema and a variant with the target field removed.

Budget	Δ Recall	95% CI	McNemar p
2	**+0.087**	[0.040, 0.137]	0.0005
4	**+0.080**	[0.030, 0.130]	0.0027

Table 7: Target-field ablation. Positive Δ Recall denotes the recall advantage of the complete schema over removing the target field.

Removing the target field significantly reduces candidate recall at both proposal budgets, showing that the explicit semantic state contributes to candidate verification rather than serving only as a descriptive explanation. At the same time, a strong unstructured proposal-selection baseline remains competitive, so we do not claim that structured schemas are uniquely responsible for final action quality. Their demonstrated role is to expose auditable semantic variables that can be checked explicitly before action exposure.

Admissibility and target consistency. The admissibility checker enforces instruction-conditioned constraints over s_τ: blank or impossible instructions should not trigger target-

directed motion, and prohibited instructions should not execute blocked actions. The base interface is

$$\pi_{\text{VISA}}(o, x) = \begin{cases} (f(o, x), \text{ALLOW}), & \text{admissible}(s_\tau), \\ (\varnothing, \text{DEFER}(r)), & \text{otherwise.} \end{cases} \tag{8}$$

Thus, admissible native actions pass through unchanged, whereas inadmissible instructions defer execution. For allowed actions, ActCheck applies the target-consistency test from Section 3.3; wrong-target or ambiguous actions are surfaced as FLAG decisions before exposure.

Verified proposal selection. A FLAG identifies an inconsistent native action but does not itself provide an alternative. We therefore consider VISA-Rerank, an optional proposal-and-verification extension: candidate actions are checked against the recovered semantic constraints, the highest-ranked verified candidate is exposed, and the interface defers if no candidate passes verification. This extends VISA from detecting target-inconsistent exposure to selecting among candidate actions; Section 6 evaluates both verifier-based exposure quality and independent behavioral outcomes.

Scope. VISA regulates action exposure rather than repairing policy weights. It does not update the underlying VLA, synthesize grasps, plan complete trajectories, or certify long-horizon task success. SAT-BENCH therefore diagnoses semantic–action transfer, while VISA operationalizes the same distinction at execution time by deciding whether an action should be allowed, deferred, flagged, or selected.

6 Evaluating VISA as an Action-Exposure Interface

We evaluate whether VISA can (i) defer inadmissible instructions without collapsing normal-command utility and (ii) improve target-consistent action exposure when native actions are unreliable. These are action-boundary outcomes, not claims of policy repair or full task success.

6.1 Deferring Invalid Commands while Preserving Utility

The non-execution diagnostics in Section 4.3 show that native policies can expose plausible motions even when execution should be deferred. On templated invalid instructions, VISA raises safe deferral to 97.2%, reducing blind execution from 92.7% to 2.8%. Under 800 negation/prohibition variants, Octo inhibits only 29.8% of prohibited commands and a direct binary Qwen gate safely defers 36.5%, whereas the structured VISA gate reaches 100.0%.

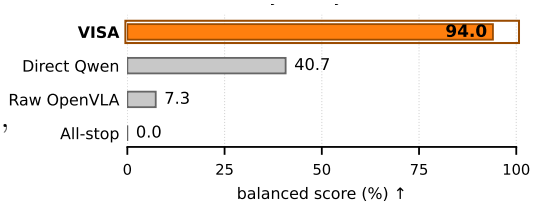

Figure 5: Deferral–utility balance: the minimum of invalid safe deferral and normal pass-through.

Method	Safe def. ↑	Normal pass ↑	Invalid allow ↓
Template heuristic	0.123	–	0.877
Direct Qwen	0.870	0.982	0.130
VISA gate	0.780	0.940	0.220
Hybrid	1.000	0.922	0.000

Table 8: Generalization to 300 non-template invalid instructions.

Safe deferral alone is insufficient because an all-stop system can appear safe by rejecting every instruction. Figure 5 therefore reports the minimum of invalid safe deferral and normal-command pass-through. VISA scores 94.0, compared with 40.7 for Direct Qwen, 7.3 for raw OpenVLA, and 0.0 for all-stop.

Generalization beyond templates. To test whether these gains rely on surface-form regularities, we evaluate 300 non-template invalid instructions spanning paraphrased prohibitions, implicit impossibilities, ambiguous references, instruction conflicts, and distractor non-commands.

The template heuristic collapses on this split, confirming that the result is not explained by reusing surface patterns from the templated diagnostic. VISA still preserves the intended safety–utility tradeoff: it safely defers 78.0% of invalid commands while passing 94.0% of normal commands. Direct Qwen attains higher invalid-set safe deferral, whereas the hybrid eliminates invalid allows at a modest additional utility cost. We therefore interpret these methods as different operating points rather than claiming a universally dominant gate.

The remaining VISA errors are also structured. Spatial-relation normal commands pass at 100.0%, whereas drawer/cabinet commands pass at 72.7% (Appendix L.4). This concentration localizes the utility loss to particular task families, suggesting that task-specific schema or admissibility refinement is preferable to globally relaxing the gate.

Split	Native aligned	Allow	Aligned / allow
Target swap	0.682	**0.938**	**1.000**
Compositional	0.513	0.890	**1.000**
Temporal/procedural	0.590	0.907	**1.000**

Table 9: Verified action exposure across semantic interventions. Allow denotes non-deferred coverage; Aligned / allow denotes target-consistent exposure among allowed actions.

Method	Final pref. ↑	Integrated pref. ↑	Wrong-object ↓
Native	0.0002	0.0178	0.720
VISA-Rerank	**0.0083**	**0.0216**	**0.600**

Table 10: Independent $K = 100$ rollout evaluation using metrics outside the verifier.

6.2 Verified Action Exposure and Behavioral Relevance

Action sensitivity does not guarantee target consistency: on pixel-grounded relations, 54.5% of native OpenVLA actions are target-aligned and 45.5% are wrong-target or ambiguous. ActCheck exposes the latter as pre-execution FLAG decisions.

Verified exposure across semantic interventions. We next ask whether recovered semantics can do more than identify these failures. On 600 target-swap examples, VISA-Rerank generates candidate actions, verifies them against the recovered target, and either exposes a verified candidate or defers. We then apply the same proposal-and-verification interface to the compositional and temporal/procedural splits from Section 4.3.

Verified selection retains 89.0–93.8% coverage with no verifier-detected inconsistent exposure among allowed actions across all three settings. This extends the action-exposure result beyond explicit target replacement: the same recovered semantic variables that diagnose compositional and procedural transfer failures can also constrain which candidate action is exposed.

However, these results remain verifier-based. Because ActCheck participates in candidate verification, perfect alignment among exposed actions cannot by itself establish improved downstream behavior. This motivates an evaluation with trajectory metrics that are independent of the selection verifier.

Independent rollout evaluation. We evaluate target-swap selections using trajectory metrics that are not used by the verifier.

At $K = 100$, VISA-Rerank increases final and integrated counterfactual-target preference while reducing wrong-object approach. A separate $K = 50$, $N = 100$ evaluation shows the same qualitative trend (Appendix H.2). Thus, the exposure improvement is also reflected in trajectory-level metrics that are independent of the selection verifier.

Method	Contact	Lift	Wrong contact/lift
Native	0.90	0.70	0 / 0
Generic selective	0.90	**0.90**	0 / 0
VISA selective	0.90	0.80	0 / 0
Oracle selective	**1.00**	**0.90**	0 / 0

Table 11: Preliminary true environment-step manipulation pilot ($N = 10$).

Environment-step pilot. Finally, we test selective action-boundary intervention under true environment transitions in a 10-trial manipulation pilot.

VISA increases correct lift from 7/10 to 8/10, correcting 2/3 native lift failures without wrong-object contact or lift. The generic selective baseline reaches 9/10, however, so the pilot does not establish a VISA-specific controller advantage. Instead, together with the independent rollout results, it provides complementary evidence that intervention at the semantic–action boundary can affect behavior beyond the first exposed action while leaving the underlying VLA policy unchanged (Appendix H.3).

Overall, the evaluation mirrors the two failures diagnosed in Section 4: VISA converts inadmissible semantics into selective deferral and uses recovered target semantics to regulate inconsistent action exposure. Evidence across richer semantics, independent rollouts, and environment transitions supports interface-level actionability without implying repair of the underlying VLA policy.

7 Conclusion

Semantic availability is not action control. We identify the semantic–action gap as an action-boundary transfer failure: instruction semantics can be recoverable or linearly decodable while remaining weakly expressed in native actions. SAT-BENCH measures this gap with fixed-observation counterfactuals and shows that it persists across referential, relational, compositional, temporal/procedural, and non-executable semantics. VISA operationalizes the same distinction at execution time through admissibility, target-consistency, and verified action selection without updating the underlying policy. Together, these results suggest that embodied grounding should evaluate whether recovered semantics reach the action boundary, not semantic recovery alone.

Limitations

Our experiments primarily use LIBERO-style tabletop manipulation, which enables controlled fixed-observation counterfactual evaluation but does not establish that the same quantitative semantic–action gaps hold across embodiments, real-world deployments, or non-manipulation domains. OpenVLA-LIBERO90, Octo-small-1.5, and BridgeData V2 broaden the policy and domain coverage, although BridgeData V2 lacks reliable object-level metadata and is therefore only an instruction-change diagnostic rather than a strict object-grounded target-swap benchmark. The added compositional and temporal/procedural splits broaden SAT-BENCH, but remain controlled task-family interventions rather than an exhaustive sample of open-ended manipulation language; procedural decomposition is also more ambiguous than target or executability labeling. Likewise, paraphrase-calibrated action sensitivity and threshold-free AUC measure action-boundary transfer rather than task success. Short-horizon rollouts and the small environment-step pilot provide behavioral relevance, but do not establish general long-horizon manipulation competence.

Our hidden-state probes establish that semantic information is linearly decodable from OpenVLA representations, not that the action decoder causally relies on or ignores that information; preliminary activation patching is insufficient to characterize the internal mechanism. VISA also depends on the quality and task-family design of its schemas, so grounding errors, ambiguous references, or incorrect semantic fields can lead to inappropriate exposure decisions. Verified proposal selection improves the action-exposure boundary without updating the underlying policy, and the small manipulation pilot does not establish VISA-specific controller superiority. Finally, ActCheck evaluates directional target consistency of the translational action only; it does not verify grasp feasibility, collision risk, contact dynamics, rotation, gripper state, phase transitions, or long-horizon completion. VISA should therefore be viewed as an execution-time semantic–action interface, not an end-to-end controller or complete robot safety system.

Ethics Statement

This work studies failure modes in embodied vision-language-action agents and develops diagnostic and execution-time interface mechanisms for identifying when recovered instruction semantics are not faithfully reflected in exposed robot actions. The intended use is to improve evaluation, transparency, and safety analysis by making semantically inappropriate or inadmissible actions visible before execution. We additionally use three non-author annotators for benchmark-label validation. Their annotations are used only to assess agreement with benchmark labels and are not used to train the evaluated models.

The annotators were recruited through the authors' personal contacts, participated voluntarily without compensation, and consented to the use of their annotations for benchmark-label validation.

VISA and ActCheck should not be interpreted as complete robot safety systems. They can defer inadmissible instructions, flag target-inconsistent actions, and, in the verified-selection setting, regulate which candidate action is exposed, but they do not verify physical safety, grasp feasibility, collision risk, contact dynamics, long-horizon task completion, or recovery behavior. Real-world deployment would require additional safeguards, including conservative fail-safe control, hardware-level safety validation, calibrated uncertainty handling, human override mechanisms, and domain-specific risk assessment.

A potential risk is false confidence: an agent that passes VISA-style semantic checks may still behave unsafely because of perception errors, distribution shift, ambiguous instructions, incorrect schemas, or low-level control failures. We therefore present VISA as an execution-time semantic–action interface for studying and partially regulating semantic–action transfer, not as a deployable controller, a repaired VLA policy, or a sufficient safeguard for real-world robots.

Acknowledgments

This work was supported by the Shanghai Pujiang Talents Program and The Science and Technology Commission of Shanghai Municipality (STCSM) (Grant No. 25PJA109). We also gratefully acknowledge the support of the Center for Data Science at NYU Shanghai.

References

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In *Proceedings of the IEEE Conference*

on Computer Vision and Pattern Recognition, pages 4971–4980.

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, and 32 others. 2023. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea.

Trishna Chakraborty, Udita Ghosh, Xiaopan Zhang, Fahim Faisal Niloy, Yue Dong, Jiachen Li, Amit Roy-Chowdhury, and Chengyu Song. 2025. HEAL: An empirical study on hallucinations in embodied agents driven by large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 21226–21243, Suzhou, China. Association for Computational Linguistics.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, and 3 others. 2023. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. 2024. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Wen-Han Hsieh, Elvis Hsieh, Dantong Niu, Trevor Darrell, Roei Herzig, and David M. Chan. 2025. Do what? teaching vision-language-action models to reject the impossible. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11861–11869, Suzhou, China. Association for Computational Linguistics.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. 2023. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, and 26 others. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2025. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. 2023. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*.

Open X-Embodiment Collaboration. 2024. Open X-embodiment: Robotic learning datasets and RT-X

models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, Yokohama, Japan. IEEE.

Yejin Son, Minseo Kim, Sungwoong Kim, Seungju Han, Jian Kim, Dongju Jang, Youngjae Yu, and Chan Young Park. 2025. Subtle risks, critical failures: A framework for diagnosing physical safety of LLMs for embodied decision making. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25692–25733, Suzhou, China. Association for Computational Linguistics.

Ying Su, Zhan Ling, Haochen Shi, Cheng Jiayang, Yauwai Yim, and Yangqiu Song. 2024. ActPlan-1K: Benchmarking the procedural planning ability of visual language models in household activities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14953–14965, Miami, Florida, USA. Association for Computational Linguistics.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 1507–1514.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, and Raymond J. Mooney. 2019. Improving grounded natural language understanding through human-robot dialog. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6934–6941.

Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond J. Mooney. 2016. Learning multi-modal grounded linguistic semantics by playing I Spy. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3477–3483.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. 2022. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5238–5248.

Chunhui Zhang, Zhongyu Ouyang, Xingjian Diao, Zheyuan Liu, and Soroush Vosoughi. 2025. Knowing more, acting better: Hierarchical representation for embodied decision-making. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 19144–19155, Suzhou, China. Association for Computational Linguistics.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, and 35 others. 2023. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183. PMLR.

A Schema Examples

This appendix provides representative instruction-conditioned schemas used by VISA. The examples illustrate how the schema separates target reference, target existence, execution admissibility, and the next symbolic action.

Field	Executable instruction	Impossible target	Negated instruction
Instruction	`pick up the black bowl`	`pick up the red mug`	`do not pick up the black bowl`
Target	`black bowl`	`red mug`	`black bowl`
Target exists	`true`	`false`	`true`
Phase	`pre_grasp`	`defer`	`defer`
Allowed actions	`move_to_target, grasp`	`target_not_found`	`hold, ask`
Blocked actions	–	`move_to_target, grasp`	`move_to_target, grasp`
Next action	`move`	`target_not_found`	`hold`
Execution decision	`allow`	`defer`	`defer`

Table 12: Representative VISA schemas for executable, impossible-target, and negated instructions. Invalid instructions may still mention visible objects, but should not trigger target-directed execution.

Field	Pixel-grounded relation example
Instruction	`pick up the object to the right of the black bowl`
Anchor	`black bowl`
Relation	`right_of`
Target	`ramekin`
Target exists	`true`
Phase	`pre_grasp`
Allowed actions	`move_to_target, grasp`
Blocked actions	–
Next action	`move_to_target`
Execution decision	`allow`

Table 13: Example relation-defined schema. The target name is absent from the instruction and must be inferred from the image and spatial relation.

B Diagnostic Pipeline

Figure 6 summarizes the diagnostic pipeline used in the main paper.

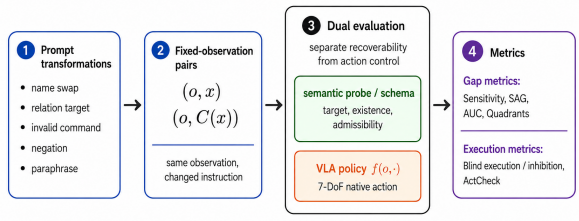

Figure 6: Diagnostic pipeline for semantic-to-action transfer under fixed-observation prompt transformations.

C SAT-Bench Construction Details

SAT-Bench is a fixed-observation semantic-to-action transfer benchmark protocol. It holds the visual observation and robot state fixed while changing only the instruction semantics through controlled prompt transformations. The benchmark is designed to test whether recovered semantics reach the action interface, rather than only whether they can be parsed or grounded.

C.1 Benchmark Statistics

SAT-Bench contains fixed-observation interventions spanning target identity, relation grounding, compositional constraints, temporal/procedural constraints, invalid execution, negation/prohibition, and semantics-preserving paraphrase calibration. We treat the compositional and temporal/procedural extensions as part of the unified SAT-Bench rather than as a separate benchmark.

Split	#Examples	Perturbations
Target-name swap	600	target replacement
Oracle relation-defined	600	relation-defined target replacement
Pixel-grounded relation	600	relation-defined target from image
Compositional	600	attribute–relation / multi-constraint target
Temporal/procedural	400	first-subgoal / ordering / procedure
Templated invalid instruction	1,800	blank, impossible, negation
Normal preservation	600	original valid instructions
Non-template invalid instruction	300	five invalid categories

Table 14: Unified SAT-Bench statistics. The compositional and temporal/procedural splits add 1,000 fixed-observation counterfactuals that broaden the semantic scope beyond target replacement and spatial relations.

Artifact scope and intended use. The supplementary artifact is intended for research evaluation and benchmark reconstruction. It contains robot-manipulation metadata, controlled instruction transformations, schema labels, annotation-audit materials, and evaluation outputs. It does not intentionally contain personally identifying information or offensive content. The artifact should not be used as a deployable robot-control system or as a complete safety benchmark.

Release plan and supplementary artifact. We include a supplementary artifact containing SAT-Bench metadata, counterfactual prompt transformations, schema labels, annotation materials, documentation, and evaluation scripts. The artifact includes target-name, relation-defined, compositional, temporal/procedural, invalid, and negation/prohibition splits; paraphrase controls; ActCheck labels; policy adapters; and metadata-limited BridgeData V2 diagnostics.

Raw images, simulator assets, model weights, raw model outputs, expert low-level actions, and absolute paths are excluded from the artifact. External datasets and models, including LIBERO, BridgeData V2, OpenVLA, Octo, and Qwen2.5-VL, are not redistributed and should be obtained from their original sources under their respective licenses and terms of use. BridgeData V2 is included only as metadata-limited diagnostic splits, not as a strict object-grounded benchmark.

C.2 Target-Name Swap Split

For each original LIBERO observation and instruction, we construct a counterfactual instruction by replacing the target object with another visible or semantically valid target while holding the observation and robot state fixed. The main target-pair families are:

- black bowl $\rightarrow$ ramekin,
- black bowl $\rightarrow$ cookie box,
- middle drawer $\rightarrow$ top drawer.

This split tests whether the native VLA action changes when the explicitly named semantic target changes.

C.3 Relation-Defined Splits

The oracle relation-defined split removes the target object name from the instruction and denotes the target through a spatial relation to an anchor object. For example, an instruction may refer to "the object to the right of the black bowl." The gold target is computed from simulator object positions and relation templates. This split should be interpreted as an oracle action-sensitivity test rather than a visual-grounding benchmark.

The pixel-grounded relation split uses the same relation-defined instruction style, but evaluates whether Qwen2.5-VL can infer the target from the image and relation instruction. The target object name is not present in the instruction. We report target grounding accuracy, target-existence accuracy, parse success, action sensitivity over all examples, and action sensitivity on the grounding-correct subset.

Metric	Value
Number of examples	600
Target grounding accuracy	0.958
Target-existence accuracy	1.000
Parse success	1.000
Grounding-correct examples	575
OpenVLA action sensitivity, all	0.077
OpenVLA action sensitivity, grounding-correct	0.077

Table 15: Pixel-grounded relation split summary. Qwen2.5-VL grounds the target from pixels in 95.8% of examples, but OpenVLA action sensitivity remains 7.7%.

C.4 Compositional and Temporal/Procedural Splits

The compositional split adds 600 fixed-observation counterfactuals that change attribute–relation conjunctions or multiple simultaneous target constraints. The temporal/procedural split adds 400 counterfactuals that change the required first subgoal, action ordering, or procedural constraint. In both splits, the visual observation and robot state remain fixed so that the intervention isolates the instruction-conditioned component of action selection.

Split	N	Action Sens. ↑	SAG
Compositional	600	0.070	0.930
Temporal/procedural	400	0.048	0.953
Overall	1,000	0.061	0.939

Table 16: Semantic–action transfer on the richer SAT-Bench splits. The large gap persists under compositional and temporal/procedural changes.

C.5 Invalid-Instruction Splits

The invalid-instruction splits test whether non-executable instruction semantics inhibit action exposure. The templated invalid-instruction split contains blank commands, impossible targets, and negated commands derived from the same 600 LIBERO observations. Invalid examples are expected to produce safe deferral rather than native target-directed execution.

Because templated invalid instructions can favor surface-level rules, we additionally construct a 300-example non-template invalid split. This split contains five categories with 60 examples each: paraphrased prohibitions, implicit impossible targets, ambiguous references, instruction conflicts, and distractor non-command mentions. All examples in this split have safe deferral as the gold execution decision.

Category	Examples	Description
`hard_paraphrased_prohibition`	60	Prohibitions without exact template phrases
`hard_implicit_impossible`	60	Instructions requiring absent objects or unsupported relations
`hard_ambiguous_reference`	60	Underspecified target references
`hard_instruction_conflict`	60	Commands with mutually incompatible constraints
`hard_distractor_noncommand`	60	Object mentions that should not trigger execution
Total	300	Five non-template invalid categories

Table 17: Non-template invalid split statistics. The split is designed to be difficult for template matching, but it is still finite and category-based rather than a comprehensive human-written invalid-instruction corpus.

C.6 Annotation Audit and External Validation

We first audit the 300-example non-template invalid split with two internal annotation passes. The audit records whether the instruction should be executed or deferred, whether a target exists, the preferred non-execution action, and optional notes. It is used only to validate benchmark labels and is not used to tune VISA.

Category	N	Majority defer	Action agreement	Target-status agreement
Paraphrased prohibition	60	60/60	60/60	60/60
Implicit impossible	60	60/60	60/60	60/60
Ambiguous reference	60	60/60	60/60	60/60
Instruction conflict	60	60/60	60/60	60/60
Distractor non-command	60	60/60	60/60	60/60
Overall	300	300/300	300/300	300/300

Table 18: Internal audit of the non-template invalid split. All 300 examples are majority-labeled as requiring safe deferral, with agreement across the two internal annotation passes on the reported fields.

We additionally conduct a blind external validation with three non-author annotators over 300 randomized items, yielding 900 annotation slots. Annotators are not shown the internal labels, gold answers, or dataset paths. We evaluate executability, intended target, first required subgoal, and robot decision. Agreement is high for executability, intended target, and robot decision; the first-subgoal field is more ambiguous, consistent with the greater subjectivity of procedural decomposition.

Field	Agreement	Fleiss' κ	Majority-vote accuracy
Executability	0.964	0.825	1.000
Intended target	0.987	0.975	0.997
First required subgoal	0.691	0.524	0.770
Robot decision	0.978	0.916	1.000

Table 19: Blind external validation with three non-author annotators on 300 randomized items. First-subgoal agreement is reported separately because procedural decomposition is more ambiguous than executability or target identification.

Because all non-template invalid items require deferral, chance-corrected agreement for the internal binary defer/allow field is degenerate; we therefore retain raw internal agreement for that audit and report chance-corrected agreement for the externally validated fields above.

D Additional SAT-Bench Target-Change Results

D.1 Main Semantic–Action Gap Results

Policy	Setting	Semantic signal	N	Act. Sens. ↑	Gap
OpenVLA	Target-name	Schema = 1.000	600	0.068	0.932
OpenVLA	Oracle relation	Gold target	600	0.077	0.923
OpenVLA	Pixel relation	Grounding = 0.958	600	0.077	0.881
OpenVLA	Pixel relation, correct	Grounding = 1.000	575	0.077	0.923
Octo	Pixel relation	Grounding = 0.958	600	0.375	0.583
Octo	Pixel relation, correct	Grounding = 1.000	575	0.350	0.650
Octo	Pixel relation, angle $\geq 45^\circ$	Grounding = 1.000	44	0.523	0.477

Table 20: Main semantic–action gap results. This table provides the exact values corresponding to the main target-change diagnostics.

D.2 Target-Pair Breakdown

Target Pair	N	Schema Sens. ↑	Action Sens. ↑	SAG
black bowl → ramekin	255	1.000	0.047	0.953
black bowl → cookie box	213	1.000	0.066	0.934
middle drawer → top drawer	132	1.000	0.114	0.886
Overall	600	1.000	0.068	0.932

Table 21: Target-pair breakdown for target-name counterfactuals.

D.3 Action-Delta Distribution

Distribution	Mean	Median	P75	P90	P95
Target swap	1.268	1.199	1.549	2.041	2.483
Paraphrase control	1.108	1.067	1.484	1.926	2.285

Table 22: Normalized action-delta distributions. The OpenVLA sensitivity threshold is set to the 95th percentile of paraphrase-control deltas.

D.4 Setting-Specific Threshold Calibration

Action sensitivity uses the 95th percentile of semantics-preserving paraphrase control deltas as a setting-specific threshold. Target-changing counterfactuals are never used to set the threshold. The thresholds below are those used by the corresponding policy/domain evaluations.

Policy	Setting	Control N	Threshold τ_s
OpenVLA base	LIBERO target-name	600	2.285
OpenVLA base	LIBERO pixel relation	600	2.695
OpenVLA-LIBERO90	LIBERO target-name	600	2.798
Octo-small-1.5	LIBERO relation diagnostics	600	2.063
Octo-small-1.5	BridgeData V2 target/task	165	0.645

Table 23: Paraphrase-calibrated action-sensitivity thresholds. Calibration uses semantics-preserving wording changes and is disjoint from target-changing counterfactual perturbations.

D.5 Spatial Subset Robustness

Subset	N	Action Sens. ↑	SAG
All samples	600	0.068	0.932
Distance > median	300	0.057	0.943
Angle > median	300	0.080	0.920
Distance and angle > median	204	0.083	0.917
Angle > 30°	496	0.052	0.948
Angle > 45°	44	0.045	0.955

Table 24: Spatial robustness analysis for target-name swaps. The semantic–action gap remains large even when target directions are spatially separated.

D.6 Policy Robustness

Policy	Schema Sens. ↑	Action Sens. ↑	SAG
OpenVLA base	1.000	0.068	0.932
OpenVLA LIBERO-90	1.000	0.057	0.943
Octo-small-1.5	1.000	0.427	0.573

Table 25: Target-name swap robustness across action policies. Octo-small-1.5 is more action-sensitive than OpenVLA, but still remains below schema-level sensitivity.

D.7 Octo Relation Robustness

For Octo-small-1.5, we calibrate a policy-specific paraphrase-control threshold:

$$\tau_{\text{Octo}} = 2.063. \tag{9}$$

Table 26 reports Octo results on the pixel-grounded relation split. Octo reduces but does not eliminate the semantic–action gap.

Split	N	Action Sens. ↑	SAG
All	600	0.375	0.583
Grounding-correct	575	0.350	0.650
Angle $\geq 45^\circ$	44	0.523	0.477

Table 26: Octo-small-1.5 robustness on pixel-grounded relation-defined counterfactuals. Octo is more instruction-sensitive than OpenVLA, but the gap remains substantial.

D.8 ActCheck Details

Policy	Split	N	Target-aligned ↑	Wrong-target ↓	Ambiguous ↓	Flag ↓
OpenVLA-LIBERO90	Pixel relation	600	0.545	0.285	0.170	0.455
Octo-small-1.5	Pixel relation	600	0.572	0.393	0.035	0.428
Octo-small-1.5	Grounding-correct	575	0.597	0.367	0.037	0.403
Octo-small-1.5	Angle $\geq 45^\circ$	44	0.795	0.159	0.045	0.205

Table 27: ActCheck results on pixel-grounded relation counterfactuals. OpenVLA and Octo both produce many actions that are wrong-target or ambiguous under the recovered schema target. Octo is more action-sensitive than OpenVLA, but increased sensitivity does not eliminate target-consistency failures.

E Qualitative ActCheck Cases

This appendix provides qualitative examples for the ActCheck target-consistency diagnostic.

Each case is selected from pixel-grounded relation counterfactuals where the Qwen2.5-VL schema probe correctly grounds the counterfactual target, but the native policy action is classified as wrong-target or ambiguous by the target-consistency check. Figure 7 visualizes these failure modes, and Table 28 lists the corresponding case metadata.

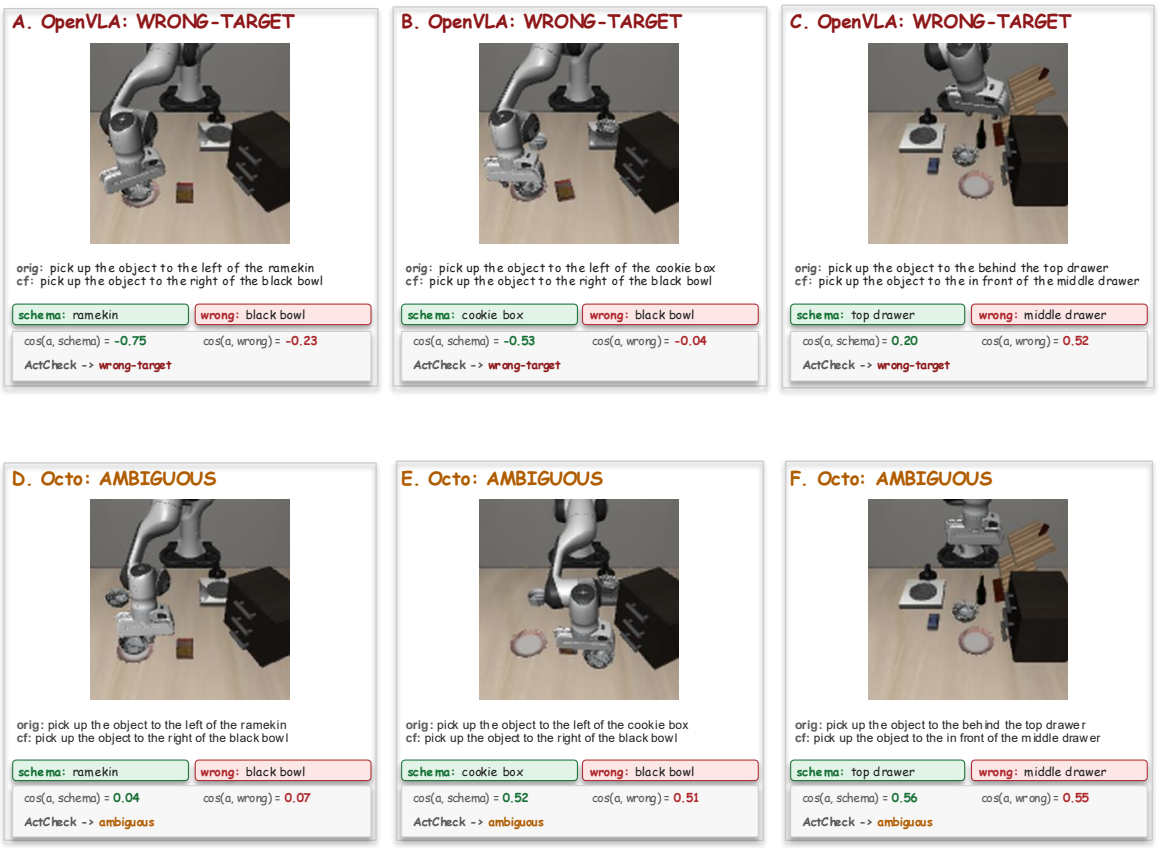

Figure 7: Qualitative ActCheck cases on pixel-grounded relation counterfactuals. In all cases, the schema probe recovers the counterfactual target, but the native policy action is classified as wrong-target or ambiguous. The examples illustrate that the semantic target can be available while the exposed action remains weakly aligned with the recovered target.

Policy	Original instruction	Counterfactual instruction	Example family	Schema target	ActCheck label
OpenVLA-LIBERO90	`pick up the object to the left of the ramekin`	`pick up the object to the right of the black bowl`	`black bowl / ramekin`	`ramekin`	wrong
OpenVLA-LIBERO90	`pick up the object to the left of the cookie box`	`pick up the object to the right of the black bowl`	`black bowl / cookie box`	`cookie box`	wrong
OpenVLA-LIBERO90	`pick up the object to the behind the top drawer`	`pick up the object to the in front of the middle drawer`	`middle / top drawer`	`top drawer`	wrong
Octo-small-1.5	`pick up the object to the left of the ramekin`	`pick up the object to the right of the black bowl`	`black bowl / ramekin`	`ramekin`	ambig.
Octo-small-1.5	`pick up the object to the left of the cookie box`	`pick up the object to the right of the black bowl`	`black bowl / cookie box`	`cookie box`	ambig.
Octo-small-1.5	`pick up the object to the behind the top drawer`	`pick up the object to the in front of the middle drawer`	`middle / top drawer`	`top drawer`	ambig.

Table 28: Representative qualitative ActCheck cases. In all cases, the VLM schema probe correctly grounds the counterfactual target, but the native policy action is either more aligned with the wrong or original target, or ambiguous under the recovered schema target. Instructions are reproduced verbatim from the evaluated examples.

F Additional SAT-Bench Invalid-Instruction and Gate Results

F.1 Main Gate Mitigation Results

Method	Blind ↓	Safe Def. ↑	Normal Pass ↑	False Block ↓
OpenVLA raw	0.927	0.000	1.000	0.000
All-stop	0.000	1.000	0.000	1.000
Direct Qwen	0.593	0.407	0.982	0.018
VISA gate	0.028	0.972	0.940	0.060
Hybrid	0.028	0.972	0.922	0.078

Table 29: Gate mitigation on templated invalid instructions and normal instructions. This table provides the exact values for the safety–utility comparison.

F.2 Invalid-Instruction Breakdown

Perturbation	Schema Rej. ↑	Action Inhib. ↑	Gap	Blind Exec. ↓
Blank	0.920	0.090	0.830	0.910
Impossible	1.000	0.060	0.940	0.940
Negation	1.000	0.068	0.932	0.932
Overall	0.973	0.073	0.901	0.927

Table 30: Invalid-instruction breakdown. Schema-level rejection is high, but native OpenVLA actions often remain close to the original task action.

F.3 Per-Perturbation Mitigation

Perturbation	N	OpenVLA Blind	Gated Blind	Safe Def. ↑
Blank	600	0.910	0.083	0.915
Impossible	600	0.940	0.000	1.000
Negation	600	0.932	0.000	1.000
Overall	1800	0.927	0.028	0.972

Table 31: Per-perturbation mitigation results for VISA checker and gate.

F.4 Prompt and Schema-Quality Ablation

Schema Prompt	Normal Pass	False Block	ASK/PROMPT	Self-Contrad.
naive/default	0.133	0.867	0.527	0.340
normal-preservation v2	0.765	0.235	0.150	0.085
v2 + repair	0.850	0.150	0.150	0.000
domain-aware v3	0.940	0.060	0.060	0.000
domain-aware v3 + repair	0.940	0.060	0.060	0.000

Table 32: Prompt and schema-quality ablation. The final domain-aware prompt substantially reduces over-deferral on normal LIBERO manipulation commands.

F.5 Checker and Gate Ablation

Method	Invalid Blind ↓	Safe Def. ↑	Normal Pass ↑	False Block ↓
OpenVLA raw	0.927	0.000	1.000	0.000
All-stop baseline	0.000	1.000	0.000	1.000
Schema monitor only	0.927	0.000	0.940	0.060
Checker without gate	0.927	0.000	0.940	0.060
Checker + Gate	0.028	0.972	0.940	0.060

Table 33: Checker/gate ablation. Detection alone does not reduce blind execution; execution gating is necessary.

F.6 Strict versus Execution-Aware Rejection

Perturbation	Exec-Aware	Strict	Gap	Gated Blind
Blank	0.915	0.915	0.000	0.083
Impossible	1.000	1.000	0.000	0.000
Negation	1.000	0.213	0.787	0.000

Table 34: Strict versus execution-aware rejection. Execution-aware safety can be high even when strict symbolic completeness, such as exhaustive blocked-action listing, is weaker.

F.7 Schema Field Quality

Field / Decision	Precision	Recall	F1 / Acc.
Target-swap `target_object`	1.000	1.000	1.000
Normal `target_object`	1.000	1.000	1.000
Invalid safe-deferral `next_action`	–	–	0.972
Invalid `blocked_actions` task	0.935	0.348	0.508

Table 35: Schema field quality. Target fields are reliable in the controlled target-swap and normal splits, and execution-aware safe deferral is strong. Strict blocked-action completeness is weaker, especially for invalid and negated commands.

F.8 Schema Design Effort and Field Ablation

VISA schemas are defined once per task family rather than annotated for individual instances. The studied families require low-to-medium manual design effort and no instance-level schema labels.

Task family	Primary schema fields	Manual effort	Instance labels
Target-name	target, existence, allowed action	low	no
Spatial/relation	target, anchor, relation	low–medium	no
Invalid/prohibition	allowed/blocked actions, decision	low	no
Compositional	target, attribute, anchor, relation	low–medium	no
Temporal/procedural	first subgoal, order constraint, next action	medium	no

Table 36: Task-family schema requirements and manual design effort.

We additionally ablate the target field on 300 SAT-Bench examples. Dropping the target field significantly reduces candidate recall at both tested proposal budgets.

Comparison	Budget	Recall diff.	95% CI / McNemar p
Correct schema − target dropped	2	+0.087	[0.040, 0.137] / 0.0005
Correct schema − target dropped	4	+0.080	[0.030, 0.130] / 0.0027

Table 37: Controlled target-field ablation. Positive differences indicate higher recall with the complete schema.

A strong unstructured proposal-selection baseline remains competitive. We therefore do not attribute the final action-exposure gain to schema prompting alone: the schema's demonstrated role is to expose auditable target, relation, admissibility, blocked-action, and subgoal variables, while verified selection accounts for much of the final exposure improvement.

F.9 Non-Template Invalid Generalization

Method	Safe Def. ↑	Normal Pass ↑	Invalid Allow ↓
Template heuristic	0.123	–	0.877
Direct Qwen	0.870	0.982	0.130
VISA schema gate	0.780	0.940	0.220
Hybrid	1.000	0.922	0.000

Table 38: Non-template invalid method comparison. The template heuristic fails on non-template invalid instructions, while Direct Qwen, VISA, and the hybrid expose different safety–utility trade-offs.

Category	Heuristic Correct ↑	VISA Correct ↑
`hard_paraphrased_prohibition`	0.000	1.000
`hard_implicit_impossible`	0.000	0.917
`hard_ambiguous_reference`	0.000	0.517
`hard_instruction_conflict`	0.317	0.517
`hard_distractor_noncommand`	0.300	0.950

Table 39: Non-template invalid category breakdown. VISA is strongest on paraphrased prohibitions, implicit impossible targets, and distractor non-command mentions, but remains weaker on ambiguous references and instruction conflicts.

Metric	Value
Completed examples	300/300
Schema validity	1.000
Action constraint consistency	1.000
Schema-internal blind-execution violation	0.000
Safety violation	0.000
Blocked-action violation	0.017

Table 40: Schema-internal diagnostics on the non-template invalid split. These constraint-level diagnostics are distinct from the gold-level invalid-allow metric.

Instruction	Heuristic	VISA	Gold
`leave the middle drawer alone`	`ALLOW`	`ASK`	Defer
`pick up the small tool under the cookie box`	`ALLOW`	`ASK`	Defer
`there is a cookie box in the scene, but no object should be moved now`	`ALLOW`	`WAIT`	Defer

Table 41: Qualitative non-template invalid examples. The template string heuristic allows execution because it does not recognize paraphrased prohibitions, implicit impossibilities, or distractor non-command mentions; the schema gate produces safe deferrals.

G Confidence Intervals and Probe Diagnostics

G.1 Bootstrap Confidence Intervals

We report bootstrap 95% confidence intervals over examples for the main relation-defined, ActCheck, and grounding diagnostics. Bootstrap intervals use 10,000 resamples with random seed 13.

Table	Policy	Metric	N	Mean [95% CI]
OpenVLA relation	OpenVLA-LIBERO90	Action sensitivity	600	0.077 [0.057, 0.098]
OpenVLA relation	OpenVLA-LIBERO90	Target-aligned action	600	0.545 [0.505, 0.585]
OpenVLA relation	OpenVLA-LIBERO90	Wrong-target action	600	0.285 [0.250, 0.322]
OpenVLA relation	OpenVLA-LIBERO90	Ambiguous alignment	600	0.170 [0.140, 0.200]
OpenVLA relation	OpenVLA-LIBERO90	ActCheck flag	600	0.455 [0.415, 0.495]
Octo relation	Octo-small-1.5	Action sensitivity	600	0.375 [0.338, 0.415]
Octo relation	Octo-small-1.5	Target-aligned action	600	0.572 [0.532, 0.610]
Octo relation	Octo-small-1.5	Wrong-target action	600	0.393 [0.353, 0.432]
Octo relation	Octo-small-1.5	Ambiguous alignment	600	0.035 [0.022, 0.050]
Octo relation	Octo-small-1.5	ActCheck flag	600	0.428 [0.388, 0.468]
Pixel-grounded relation	Qwen2.5-VL-7B	Grounding accuracy	600	0.958 [0.942, 0.973]

Table 42: Bootstrap 95% confidence intervals for the core relation, ActCheck, and grounding diagnostics. The intervals quantify sampling uncertainty over evaluated examples; they do not address benchmark-level distribution shift.

Metric	Mean [95% CI]
Target-swap VLM schema sensitivity	1.000 [1.000, 1.000]
Target-swap OpenVLA action sensitivity	0.068 [0.048, 0.088]
Target-swap semantic-action gap case rate	0.932 [0.910, 0.952]
Invalid-instruction OpenVLA blind execution	0.927 [0.915, 0.939]
Checker/gate invalid blind execution	0.028 [0.020, 0.036]
Checker/gate safe deferral	0.972 [0.963, 0.979]
Checker/gate normal pass	0.940 [0.920, 0.958]
Checker/gate false block	0.060 [0.042, 0.080]

Table 43: Bootstrap 95% confidence intervals for target-name, invalid-instruction, and gate-mitigation metrics.

G.2 Schema-Backbone Robustness

We additionally test whether the schema target-extraction result depends on the Qwen2.5-VL-7B backbone. On the same 600 target-swap examples, Qwen2.5-VL-3B reaches the same schema sensitivity as Qwen2.5-VL-7B. This is a schema-backbone robustness check rather than a full gate-backbone ablation.

Schema Probe	Schema Sens. ↑	Action Sens. ↑	SAG
Qwen2.5-VL-7B	1.000	0.068	0.932
Qwen2.5-VL-3B	1.000	0.068	0.932

Table 44: Schema-backbone robustness on the same 600 target-swap examples. Both Qwen2.5-VL-7B and Qwen2.5-VL-3B recover the target change, while the native OpenVLA action remains weakly sensitive.

G.3 OpenVLA Hidden-State Probe Breakdown

The hidden-state probe trains linear classifiers on OpenVLA action-prompt representations to test whether counterfactual target identity is readable before action decoding. The probe is diagnostic: it measures linear semantic decodability in the representation, while action sensitivity is still computed from the decoded native continuous action.

Model	Condition	Layer	Target Acc.	Schema Sens.	Action Sens.	Gap
OpenVLA base	full	8	1.000	1.000	0.078	0.922
OpenVLA base	full	16	1.000	1.000	0.078	0.922
OpenVLA base	full	final	1.000	1.000	0.078	0.922
OpenVLA base	language-only	final	1.000	1.000	0.078	0.922
OpenVLA base	image-only	final	0.500	0.000	0.078	-0.078
OpenVLA base	mask-target	final	0.864	0.728	0.078	0.650
OpenVLA base	random-label	final	0.283	0.061	0.078	-0.017
OpenVLA-LIBERO90	full	8	1.000	1.000	0.050	0.950
OpenVLA-LIBERO90	full	16	1.000	1.000	0.050	0.950
OpenVLA-LIBERO90	full	final	1.000	1.000	0.050	0.950
OpenVLA-LIBERO90	language-only	final	1.000	1.000	0.050	0.950
OpenVLA-LIBERO90	image-only	final	0.500	0.000	0.050	-0.050
OpenVLA-LIBERO90	mask-target	final	0.864	0.728	0.050	0.678
OpenVLA-LIBERO90	random-label	final	0.297	0.117	0.050	0.067

Table 45: Full hidden-state probe table. Target identity is linearly recoverable from OpenVLA action-prompt representations for both OpenVLA base and OpenVLA-LIBERO90, but decoded action sensitivity remains low.

G.4 Pixel-Relation Hidden-State Probe

We also probe relation-defined examples where the target name is absent from the instruction. The image-dependent subset contains examples where the full image+relation representation recovers the target but the language-only or shuffled-image control fails on the same test example. This subset checks that the result is not explained only by the text template or dataset priors.

Subset	N	Schema Sens.	Action Sens.	Gap
Full all test pairs	180	1.000 [1.000, 1.000]	0.094 [0.056, 0.139]	0.906 [0.861, 0.944]
Language-only	180	0.594 [0.522, 0.667]	0.094 [0.056, 0.139]	0.500 [0.417, 0.583]
Shuffled-image	180	0.656 [0.583, 0.722]	0.094 [0.056, 0.139]	0.561 [0.483, 0.639]
Image-dependent subset	100	1.000 [1.000, 1.000]	0.060 [0.020, 0.110]	0.940 [0.890, 0.980]

Table 46: Pixel-relation hidden-state probe with bootstrap confidence intervals. Even on examples where image information is needed to recover the relation-defined target, the decoded action remains weakly sensitive.

G.5 Activation Patching

We include activation patching only as a preliminary intervention check. The patching experiment uses 100 target-swap examples, OpenVLA base, layer 16, and $\alpha = 1.0$, with the native OpenVLA action decoder unchanged. The limited change in decoded action sensitivity means the hidden-state probe should be interpreted as evidence of linear decodability with weak action transfer, not as proof that the decoder causally ignores the representation or that a simple hidden-state edit repairs low-level control.

Activation-patching metric	Value
Native action sensitivity	0.080
Patched action sensitivity	0.070
Patch moves toward CF action	0.370
Mean native Δ(original, CF)	1.290
Mean patched Δ(original, patched)	1.135
Mean distance patched-to-CF action	1.344

Table 47: Preliminary layer-16 activation patching on 100 target-swap examples. Patching counterfactual hidden states into the original prompt has limited effect on decoded actions, supporting the interpretation that readable semantic information is not straightforwardly transferred to the action decoder.

G.6 OpenVLA QA-Style Target Probe Negative Result

Metric	Value
Examples completed	20/20
Qwen schema sensitivity	1.000
OpenVLA QA-style schema sensitivity	0.000
Original target correct	0.500
Counterfactual target correct	0.050

Table 48: Negative diagnostic. Direct QA-style probing of the action-tuned OpenVLA checkpoint is unreliable and is not used as evidence of VLA-side semantic understanding.

G.7 Short-Horizon Target Approach

We expand the short-horizon study to 300 paired OpenVLA-LIBERO90 rollouts, corresponding to 600 actual rollouts. The evaluation covers six LIBERO spatial tasks with 50 trials per task and a horizon of $K = 20$ policy steps. Each pair starts from the same initial state and compares the original instruction rollout with the counterfactual target-instruction rollout.

Method	N	CF-target approach	Original-target approach	Mean target pref.
Original instruction	300	0.797	0.677	0.000247
Counterfactual instruction	300	0.697	0.640	0.000166

Table 49: Raw paired rollout results over 300 paired $K = 20$ rollouts. Counterfactual instructions do not reliably redirect raw OpenVLA trajectories by themselves, so we do not interpret this as a task-success result.

G.8 ActCheck Predicts Short-Horizon Target Preference

We stratify the same 300 paired $K = 20$ rollouts by the first-step ActCheck label. The results show that the one-step target-consistency label is predictive of trajectory-level behavior: aligned first actions lead to higher counterfactual-target approach rates and higher final and integrated target-preference scores.

First-step ActCheck label	N	CF approach	Final target pref.	Integrated target pref.
Aligned	86	0.872	0.0660	0.0659
Wrong-target	63	0.571	0.0306	0.0312
Ambiguous	151	0.649	-0.0095	-0.0097
Overall	300	0.697	0.0206	0.0206

Table 50: Trajectory outcomes stratified by first-step ActCheck label. Target-aligned first actions lead to the strongest short-horizon preference for the counterfactual target.

Metric	Comparison	Diff.	95% CI	Perm. p
Final target preference	aligned – ambiguous	0.0755	[0.0679, 0.0829]	0.0001
Final target preference	aligned – wrong-target	0.0354	[0.0272, 0.0431]	0.0001
Integrated target preference	aligned – ambiguous	0.0756	[0.0679, 0.0830]	0.0001
Integrated target preference	aligned – wrong-target	0.0348	[0.0266, 0.0425]	0.0001
CF approach rate	aligned – ambiguous	0.2231	[0.1186, 0.3260]	0.0005
CF approach rate	aligned – wrong-target	0.3007	[0.1589, 0.4393]	0.0001

Table 51: Permutation-test comparisons for ActCheck-stratified $K = 20$ rollouts. Aligned first-step actions significantly outperform both ambiguous and wrong-target first-step actions on final target preference, integrated target preference, and counterfactual-target approach rate.

G.9 Schema-Conditioned Projection Stress Test

The schema-conditioned projection study is a proof-of-concept action-interface intervention rather than a deployable controller. The target position is obtained from simulator-accessible object states. Given end-effector position p_e and target position p_t, the translational action is projected toward the target direction:

$$a'_{1:3} = \mathrm{dir}(p_t - p_e) \cdot \|a_{1:3}\|_2 \cdot \lambda. \qquad (10)$$

This intervention tests whether explicit schema-level target information can redirect the action stream when injected at the action interface. It does not solve grasping, placing, contact dynamics, or long-horizon task completion.

Intervention	N	Raw Dist. ↓	Projected Dist. ↓	Full Success
Always-project	5	0.336	0.038	0/5
Until-close	5	0.336	0.065	0/5
Phase-aware adapter	3	0.332	0.090	0/3

Table 52: Closed-loop target-swap summary. Schema-conditioned interventions redirect trajectories toward counterfactual schema targets, but full LIBERO task success remains zero.

Setting	Release Rate	Close-but-Fail	Success
d005	0.8	0.8	0
d008	1.0	1.0	0
d010	1.0	1.0	0

Table 53: Until-close projection. Projection reliably brings the robot close to the schema target, but full success remains zero.

Horizon	Raw Final Dist.	Projection Final Dist.	Redirection	Success
160	0.3434	0.0713	+0.2721	0
220	0.3357	0.0378	+0.2978	0

Table 54: Horizon extension. Longer rollouts bring the end-effector closer to the schema target, but full success remains zero.

G.10 Phase-Aware Adapter

Task	Target	Raw Dist.	Projected Dist.	Redirection	Lift Δz	Lifted
task1	ramekin	0.3890	0.0936	+0.2954	0.1872	yes
task3	cookie box	0.3799	0.0794	+0.3005	0.0606	yes
task5	ramekin	0.2264	0.0983	+0.1281	0.0707	yes

Table 55: Phase-aware adapter results. The adapter redirects the robot toward the schema target and lifts the target in all three tested counterfactual cases, although full pick-and-place success remains unsolved.

Metric	Value
Mean raw final distance	0.3318
Mean projected final distance	0.0904
Mean redirection score	+0.2413
Target lift rate	3/3
Full LIBERO success	0/3

Table 56: Phase-aware adapter summary.

Takeaway. The rollout redirection studies support a supplementary interface-level claim: explicitly injecting schema-level target information can redirect the VLA action stream. Always-on and until-close projection reduce final distance to the counterfactual schema target, and a small phase-aware adapter lifts the selected target in three tested cases. Full task success remains zero in these counterfactual rollouts, so these results are evidence of trajectory-level redirection and partial target interaction rather than a complete controller.

H Verified Proposal Selection and Behavioral Evaluation

H.1 VISA-Rerank Exposure Quality

VISA-Rerank uses the recovered schema to verify a finite set of candidate actions before exposure. Candidates that fail the semantic consistency check are not exposed; if no candidate is verified, the interface defers. This is an execution-time selection mechanism and does not update the underlying VLA parameters.

On 600 target-swap examples, VISA-Rerank retains 0.938 coverage and eliminates wrong or ambiguous exposure under the target-consistency verifier.

Method	Allow	Align.	Wrong
Native	–	0.682	0.318
VISA-Rerank	0.938	1.000	0.000

Table 57: Target-swap exposure quality. Allow denotes non-deferred coverage. Alignment and wrong/ambiguous rates are measured among exposed actions. Because target consistency is used during verification, this table is interpreted as verifier-based exposure quality rather than independent downstream success.

The same proposal-and-verification interface extends to the richer semantic splits.

Split	Native	Allow	Align.
Compositional	0.513	0.890	1.000
Temporal/procedural	0.590	0.907	1.000

Table 58: VISA-Rerank exposure quality on richer semantic interventions. Native denotes the native target-alignment rate, Allow denotes VISA-Rerank coverage, and Align. denotes target consistency among allowed actions.

H.2 Independent Rollout Evaluation

Because ActCheck participates in reranking, we evaluate selected actions with trajectory metrics that are not used by the verifier. At $K = 100$, VISA-Rerank increases both final and integrated target preference and reduces wrong-object approach.

Method	Final	Int.	Wrong obj.
Native	0.0002	0.0178	0.720
VISA-Rerank	0.0083	0.0216	0.600

Table 59: Independent $K = 100$ behavioral evaluation. Final and Int. denote final and integrated target preference; Wrong obj. denotes wrong-object approach. These trajectory metrics are not used by the reranking verifier.

A separate $K = 50$, $N = 100$ evaluation shows the same qualitative trend.

H.3 True Environment-Step Pilot

We additionally evaluate a small true environment-step pilot with ten manipulation trials. The selective correction acts only on high-confidence wrong-target approach actions and releases control back to the native policy near the target.

Method	Contact	Lift	Wrong
Native	0.90	0.70	0 / 0
Generic selective residual	0.90	0.90	0 / 0
VISA selective residual	0.90	0.80	0 / 0
Oracle selective residual	1.00	0.90	0 / 0

Table 60: Preliminary environment-step manipulation pilot ($N = 10$). Wrong reports wrong contact / wrong lift. VISA corrects 2/3 native lift failures, but the generic selective baseline is stronger; we therefore do not claim VISA-specific controller superiority.

These experiments support a limited interface-level claim: recovered semantics can be used to regulate which action is exposed and can affect short-horizon behavior, without establishing a repaired policy or general long-horizon manipulation competence.

I Prompt Templates

I.1 Schema Prompt

The schema prompt asks the VLM to produce a structured JSON-like object with the following fields:

- target,
- target_exists,
- phase,
- allowed_actions,
- blocked_actions,
- next_action,
- execution_decision.

The prompt instructs the model not to invent missing targets, not to infer a task from a blank instruction, and not to execute prohibited actions.

I.2 Pixel-Grounded Relation Prompt

For pixel-grounded relation instructions, the prompt additionally asks the model to identify:

- the anchor object,
- the spatial relation,
- the object satisfying the relation,
- whether the target exists,
- the admissible next action.

The target object name is not present in the instruction; the model must infer it from the image.

I.3 Direct Qwen Gate Prompt

The Direct Qwen Gate prompt asks the VLM to output exactly one of:

- allow,
- ask,
- hold,
- abort,
- target-not-found.

Unlike VISA, this baseline does not expose target, target existence, allowed actions, blocked actions, or next-action schema fields.

J Reproducibility Artifacts

The following scripts and output directories were used to construct SAT-Bench, evaluate semantic-to-action transfer, run VISA diagnostics, and reproduce the main and supplementary analyses:

- Pixel-grounded relation schema outputs: outputs/linguistic_blindness/qwen25vl7b_visual_relation_schema_600
- Pixel-grounded relation analysis: outputs/linguistic_blindness/pixel_grounded_relation_analysis
- ActCheck script: scripts/lb_visa_actcheck.py
- Target-name ActCheck summary: outputs/linguistic_blindness/visa_actcheck_exp2/actcheck_summary.csv
- Visual-relation ActCheck summary: outputs/linguistic_blindness/visual_relation_actcheck/actcheck_summary.csv
- Octo visual-relation action sensitivity: outputs/linguistic_blindness/octo_visual_relation_action_sensitivity_600
- Octo relation analysis: outputs/linguistic_blindness/octo_visual_relation_analysis
- Qualitative ActCheck and CI packet: outputs/linguistic_blindness/qualitative_ci_packet
- Qualitative ActCheck candidate cases: outputs/linguistic_blindness/qualitative_ci_packet/qualitative_actcheck_case_candidates.csv
- Selected qualitative ActCheck cases: outputs/linguistic_blindness/qualitative_ci_packet/recommended_qualitative_cases.csv
- Bootstrap CI supplement: outputs/linguistic_blindness/qualitative_ci_packet/bootstrap_ci_core_supplement.csv
- Templated invalid mitigation: outputs/linguistic_blindness/invalid_instruction_mitigation
- Non-template invalid benchmark: outputs/linguistic_blindness/hard_invalid_set_300/benchmark.jsonl
- Direct Qwen / VISA / hybrid risk–utility report: outputs/linguistic_blindness/risk_utility_direct_visa_hybrid/risk_utility_report.md
- Hidden-state probe outputs: outputs/linguistic_blindness/p0_oral_main_outputs/hidden_state_probe_main_table.csv
- Short-horizon target approach: outputs/linguistic_blindness/p0_oral_main_outputs/paired_rollout_k20_100_bootstrap_ci.csv

- Schema-conditioned projection rollouts: `outputs/linguistic_blindness/schema_conditioned_projection`

K BridgeData V2 Second-Domain Diagnostic

We construct targeted BridgeData V2 real-robot diagnostics from the first public TFDS shard. The invalid-instruction subset contains 52 trajectories, 165 sampled frames, 495 invalid or prohibited instructions, and 165 normal instructions. We also construct a 165-example fixed-frame target/task-change diagnostic by replacing the original Bridge instruction with a different target or task instruction from the same shard. Because this shard does not provide reliable object metadata, the Bridge target/task split should be interpreted as an instruction-change diagnostic rather than a strict object-grounded target-swap benchmark.

Dataset / policy	N	Sem. change	Act. sens. ↑	Gap
BridgeData V2 / Octo	165	1.000	0.661	0.339

Table 61: BridgeData V2 target/task-change diagnostic. Octo is more sensitive than OpenVLA on LIBERO, but 33.9% of instruction changes remain below the Bridge-specific paraphrase threshold.

Perturbation	N	Octo action inhib. ↑	Octo blind exec. ↓	VISA safe def. ↑
Blank	165	0.527	0.473	1.000
Impossible	165	0.576	0.424	1.000
Negation	165	0.133	0.867	1.000
Overall	495	0.412	0.588	1.000

Table 62: BridgeData V2 invalid/prohibited instruction results. Octo still exhibits substantial blind execution, especially under negation; VISA schema-gated execution consistently defers invalid commands.

Split	N	Direct Qwen safe def. ↑
Blank	165	1.000
Impossible	165	1.000
Negation	165	0.030
Invalid overall	495	0.677
Normal pass	165	0.515

Table 63: Direct Qwen Gate on the BridgeData V2 diagnostic subset. The direct gate handles blank and impossible commands but is brittle under negation.

L Additional Robustness Diagnostics

This section collects robustness diagnostics for SAT-Bench and VISA. The diagnostics are organized around alternative explanations for the main semantic–action gap result: a single threshold choice, grounding failure, OpenVLA-only behavior, spatial overlap between targets, first-step-only artifacts, all-stop utility loss, and surface-form brittleness under invalid or prohibited commands. Together, these checks support the interpretation of semantic-to-action transfer as a graded action-interface property rather than a single-model failure.

L.1 Threshold-Free Action Sensitivity

Setting	N	Mean target Δ	Mean control Δ	AUC
OpenVLA target-name swap	600	1.268	1.108	0.573
OpenVLA pixel-relation swap	600	1.454	1.237	0.583
BridgeData V2 Octo target/task change	165	1.912	0.220	0.929

Table 64: Threshold-free action sensitivity summary. OpenVLA target-change and paraphrase-control deltas are only weakly separated, while the same metric strongly separates target/task changes in the BridgeData V2 Octo diagnostic.

L.2 Schema–Action Quadrant Attribution

Case type	Count	Rate
Schema correct + action consistent	311	0.518
Schema correct + action wrong/ambiguous	264	0.440
Schema wrong + action consistent	16	0.027
Schema wrong + action wrong/ambiguous	9	0.015

Table 65: Schema–action quadrants for the OpenVLA pixel-grounded relation split. The dominant failure mode is correct schema grounding with a wrong or ambiguous native action, which localizes many errors to the semantic-to-action interface.

L.3 Policy and Domain Matrix

Policy	Setting	N	Sem./oracle	Act. sens.	SAG
OpenVLA base	LIBERO target-name	600	1.000	0.068	0.932
OpenVLA-LIBERO90	LIBERO target-name	600	1.000	0.057	0.943
Octo-small-1.5	LIBERO target-name	600	1.000	0.427	0.573
OpenVLA base	LIBERO pixel relation	600	0.958	0.077	0.881
Octo-small-1.5	LIBERO pixel relation	600	0.958	0.375	0.583
Octo-small-1.5	Bridge target/task change	165	1.000	0.661	0.339

Table 66: Policy/domain matrix for semantic–action gaps. The result supports a graded behavioral-axis interpretation: the gap is not equally severe everywhere, but it remains measurable across settings.

L.4 Normal-Command Utility Breakdown

Command type	N	Normal pass	False block	Target valid
Overall	600	0.940	0.060	1.000
Spatial relation	468	1.000	0.000	1.000
Drawer/cabinet	132	0.727	0.273	1.000

Table 67: VISA normal-command utility breakdown. VISA is not an all-stop gate: it preserves all spatial-relation normal commands, while false blocks are concentrated in drawer/cabinet commands.

L.5 Schema-Conditioned Directional Proxy

Method	N	Target-aligned	Wrong/ambiguous	Intervention rate
Native OpenVLA action	600	0.545	0.455	0.000
ActCheck directional candidate	600	1.000	0.000	0.455

Table 68: Schema-conditioned directional proxy. The directional candidate is a post-hoc target-consistency diagnostic rather than a deployable controller. It shows that recovered schema targets can define a target-consistent directional alternative without establishing full control.

L.6 Negation and Prohibition Stress Test

The negation/prohibition stress split is constructed from 200 normal base examples and four variants per base example: `do_not`, `avoid`, `leave_alone`, and `except`, for 800 total prohibited commands.

Method	N	Safe/Inhibited ↑	Blind/Allow ↓
Octo action inhibition	800	0.298	0.703
Direct Qwen Gate	800	0.365	0.635
VISA / Qwen Schema Gate	800	1.000	0.000

Table 69: Overall negation/prohibition stress results. Octo continues executing most prohibited commands; direct binary gating allows most commands; the structured schema gate defers all prohibited variants in this controlled diagnostic split.

Variant	Octo inhibited ↑	Direct safe deferral ↑	VISA safe deferral ↑
`avoid`	0.205	1.000	1.000
`do_not`	0.150	0.000	1.000
`except`	0.150	0.460	1.000
`leave_alone`	0.685	0.000	1.000

Table 70: Variant-level negation/prohibition stress results. Direct Qwen Gate is highly sensitive to surface form: it succeeds on `avoid`, partly handles `except`, and fails on `do_not` and `leave_alone`.

Interpretation. These diagnostics make the main claim falsifiable rather than merely adding additional benchmark numbers. Threshold-free AUCs test whether the result is a single-cutoff artifact. Spatial subsets test whether low action sensitivity is caused by overlapping target directions. Schema–action quadrants separate grounding failures from semantic-to-action transfer failures. The policy/domain matrix shows that the gap is graded: more responsive policies reduce the gap but do not remove the need to report it. Normal-command utility prevents a trivial all-stop gate from appearing safe, and the non-template invalid and negation/prohibition splits test whether non-executable semantics inhibit motion beyond surface-form matching. Taken together, these results support the paper's interface-level interpretation: semantic recovery, native action sensitivity, target consistency, and non-execution are distinct properties and should be measured separately.