VLA Grounder: Language-Conditioning Space Optimization for Black-Box VLA Models

Anonymous ACL submission

Abstract

Vision-Language-Action (VLA) models are commonly treated as end-to-end action policies conditioned on natural-language task descriptions. In practice, however, their behavior often depends sharply on how the instruction is phrased, suggesting that language is not merely a task label but an optimizable conditioning input. We study whether frozen VLA policies can be improved by optimizing language space rather than updating action weights. Our method introduces a language-conditioning space policy that translates a human instruction into a short VLA-grounded command using object appearance, spatial relations, and target-grounding cues. The language-conditioning space policy is initialized with a failure-derived command-space prior and optimized with reinforcement learning from sparse task-completion rewards, while the downstream VLA remains fully frozen. This yields language-conditioning space optimization: RL discovers which VLA-grounded commands best elicit successful behavior from the frozen action policy. Experiments on RL4VLA and VL-Think show that language-conditioning space optimization improves success on instruction-sensitive, symbolic, and multi-object manipulation tasks, demonstrating that language can serve as an optimizable variable for a robot foundation models.

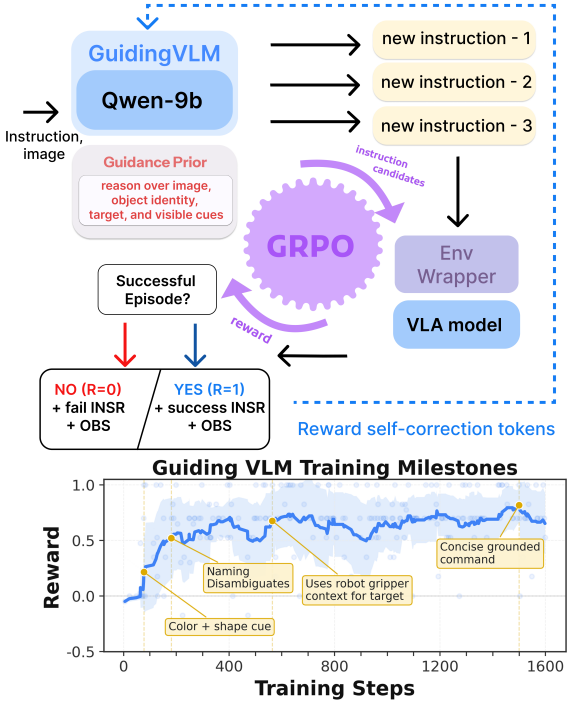

Figure 1: Visual overview of language-conditioning space optimization. A failure-derived command-space prior conditions a pre-trained VLM to translate the user instruction and scene image into a short VLA-grounded command. This command conditions a VLA policy, improving action success without updating VLA weights.

1 Introduction

Vision-Language-Action (VLA) models represent a promising paradigm in robotic learning, combining visual perception, natural language understanding, and action generation within a single policy. Recent VLA models leverage large-scale pre-training and robotic demonstrations to produce actions directly from images and language, enabling broad zero-shot manipulation across tasks and embodiments (Black et al., 2024; Zhang et al., 2025; Zhong et al., 2025).

Despite this progress, the language input to a VLA is often treated as a passive task description, even though it directly shapes the action distribution of the policy. In practice, VLA behavior is highly sensitive to how a goal is expressed (Pugacheva et al., 2025; Kachaev et al., 2025). A command such as *"put baguette on plate"* may fail even when the scene contains a visually clear target, while a more grounded command such as *"pick up the brown round object and put it on the yellow plate"* can succeed. Similar effects appear when an object name is visually misleading: *"put champagne glass on plate"* can be improved by refer-

ring to the same object as *"the tall white glass"*. These examples suggest that many VLA failures arise from an instruction-to-grounding mismatch between human semantic intent and the perceptual categories available to the robot policy.

This mismatch is amplified by the limited linguistic diversity of robot demonstration datasets (Walke et al., 2024; Collaboration et al., 2025). Most datasets contain short templated instructions and everyday household objects, while benchmark tasks may include ambiguous objects, abstract references, uncommon categories, or multiple visually similar distractors. As a result, the language channel of a VLA is underused: models reserve substantial capacity for text, but the instruction often fails to provide the visual and spatial cues needed for robust action selection (Zhang et al., 2025; Liu et al., 2025; Black et al., 2024).

A natural response is to modify the instruction rather than the VLA weights. We therefore propose to insert an additional language-conditioning space policy between the human instruction and the frozen VLA, and ask whether command selection can provide an effective adaptation mechanism for large robot foundation models without updating their action weights. Given the image and the original instruction, this policy translates the user's goal into a short VLA-grounded command that is better aligned with the perceptual and linguistic priors of the downstream VLA. Under this view, language is not a fixed task label, but an input space in which human intent is converted into an executable command for the robot policy.

We instantiate this idea with a language-conditioning space policy placed upstream of a frozen VLA, as shown in Figure 1. Given the current image and the original human instruction, the language-conditioning space policy produces a short VLA-grounded command that exposes execution-relevant cues: object appearance, color, shape, spatial position, relative location, and simplified target descriptions. The generated command is then used as the language input to the unchanged VLA policy. Because the downstream action model is kept fixed, the method adapts the conditioning signal rather than the action policy, making it non-invasive for different frozen VLA models.

We train the language-conditioning space policy (VLA Grounder) with Reinforcement Learning (RL) from sparse task-completion rewards while keeping the downstream VLA fully frozen (Figure 1)). This casts adaptation as *language-conditioning space optimization*: the learned decision variable is not a continuous robot action or a VLA weight update, but the VLA-grounded command through which the frozen action model receives and grounds the task. To make this optimization meaningful, we use a failure-derived command-space prior that restricts exploration to concise, visually grounded, and VLA-compatible commands. Within this structured command space, RL provides an action-grounded signal for discovering which commands most reliably elicit successful behavior from the existing policy. Experiments on VL-Think (Kachaev et al., 2025) and RL4VLA (Liu et al., 2025) benchmarks show that our method creates better grounded guidance commands for frozen π_0 and OpenVLA models, increasing downstream task success across object grounding and multi-object manipulation settings.

The contributions of this work are as follows:

1. **Language-conditioning space optimization for VLA models**: We formulate VLA adaptation as learning a language-conditioning space policy that maps human instructions to VLA-grounded commands while keeping the downstream action policy fixed.

2. **RL in language-conditioning space**: We introduce VLA Grounder, language-conditioning space policy that maps an image and human instruction to a concise VLA-grounded command. We optimize only the language-conditioning space policy from sparse task-completion rewards, showing that downstream robot behavior can improve without updating VLA action weights.

3. **Empirical and diagnostic evidence**: We evaluate on VL-Think and RL4VLA with frozen π_0 and OpenVLA backbones. The results show improved success on symbolic grounding, multi-object manipulation, and fixed-object settings, and our analyses indicate that optimized commands provide more grounded conditioning signals for the downstream VLA.

2 Related Work

2.1 Language-Conditioned VLA Policies

Large-scale robot policies increasingly separate high-level semantic control from low-level motor generation. ACT introduced action chunking for manipulation (Zhao et al., 2023), RT-2 transferred

web-scale vision-language pretraining into robot action generation (Brohan et al., 2023), and recent generalist policies such as OpenVLA and π_0 scale this recipe to broad zero-shot manipulation (Kim et al., 2024; Black et al., 2024). Embodied chain-of-thought and $\pi_{0.5}$ further push toward richer intermediate reasoning and semantic control over action generation (Zawalski et al., 2025; Intelligence et al., 2025b). Another line of work makes a VLA more controllable by enriching its conditioning signal. Higher-level semantic prediction in $\pi_{0.5}$, semantic guidance methods for VLA robustness, and prompt optimization methods in vision-language settings all support the broader idea that behavior depends strongly on how context is represented (Intelligence et al., 2025b; Zhan et al., 2026; Lee et al., 2023). These approaches treat language, metadata, or intermediate semantic variables as mechanisms for conditioning behavior.

Our paper asks a complementary post-training question: given an already frozen VLA, can we learn a better VLA-grounded command for control? Rather than changing the downstream action policy, we optimize the upstream language-conditioning space policy that conditions it.

2.2 RL Adaptation of VLA Models

Reinforcement learning has become a central tool for improving VLA policies after imitation learning. RL4VLA studies PPO (Schulman et al., 2017)-, GRPO (Shao et al., 2024)-, and preference-based fine-tuning for VLA generalization (Liu et al., 2025); π_{RL} extends online RL to flow-based VLA models (Chen et al., 2026); and $\pi^*_{0.6}$ shows how deployment and corrective data can substantially improve a VLA through continued learning (Intelligence et al., 2025a). More broadly, reward optimization for language policies, including ArCHer, PReWrite, and StablePrompt, shows that sparse reward can shape text-producing policies in useful ways (Zhou et al., 2024; Kong et al., 2024; Kwon et al., 2024). In contrast, our method adapts the language policy itself, not the VLA model weights, enabling lightweight post-training in language-conditioning space.

2.3 VLA Language Robustness

Recent work shows that language remains an underdeveloped modality in embodied AI. Dataset audits find highly repetitive and template-like instructions in robot corpora (Wanna et al., 2026). Stable Language Guidance frames related failures as modality imbalance, where strong visual priors overpower sparse linguistic signals (Zhan et al., 2026), adversarial paraphrases and irrelevant context can significantly degrade VLA behavior even when task intent is preserved (Pugacheva et al., 2025; Zinkovich et al., 2025), and mechanistic analyses suggest that language-conditioned behavior is present internally but not always reliably exposed by the input instruction (Häon et al., 2025). We build on this diagnosis but reframe it as an optimization problem over command space. Instead of treating language brittleness as only a robustness issue, we treat language conditioning as an actionable input space that can be optimized for downstream execution success.

3 Problem Formulation

3.1 Partially Observable Control with Language Instructions

We consider an episodic partially observable Markov decision process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, Z, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the latent state space, $\mathcal{A}$ the primitive action space, $\mathcal{O}$ the observation space, P the transition kernel, Z the observation kernel, r the reward function, ρ_0 the initial-state distribution, and $\gamma \in [0, 1)$ the discount factor. At time t, the environment occupies state s_t, emits $o_t \sim Z(\cdot \mid s_t)$, receives action a_t, produces reward $r(s_t, a_t)$, and transitions to $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. Each episode is paired with a fixed human instruction $u \in \mathcal{U}_{\text{H}}$. We write the discounted return of a trajectory τ as $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$.

3.2 Frozen Vision-Language-Action Policies

A pretrained Vision-Language-Action policy maps an observation and a language command to an action distribution, $\pi_{\text{VLA}} : \mathcal{O} \times \mathcal{U} \to \Delta(\mathcal{A})$, where $\mathcal{U}$ is the space of commands consumed by the action policy. In standard deployment, the raw instruction is passed directly to the policy, so $a_t \sim \pi_{\text{VLA}}(\cdot \mid o_t, u)$. More generally, each command $\tilde{u} \in \mathcal{U}$ induces a frozen low-level policy $\pi^{\tilde{u}}_{\text{VLA}}(a \mid o) := \pi_{\text{VLA}}(a \mid o, \tilde{u})$. Distinct commands that preserve the same task intent may still induce different grounded behavior.

Let $\mathcal{U}(u) \subseteq \mathcal{U}$ denote the command space associated with instruction u, i.e., the set of commands that are semantically consistent with u but may differ in wording, specificity, or grounding cues. The goal is not to learn a new low-level action policy, but to exploit this command space to obtain better

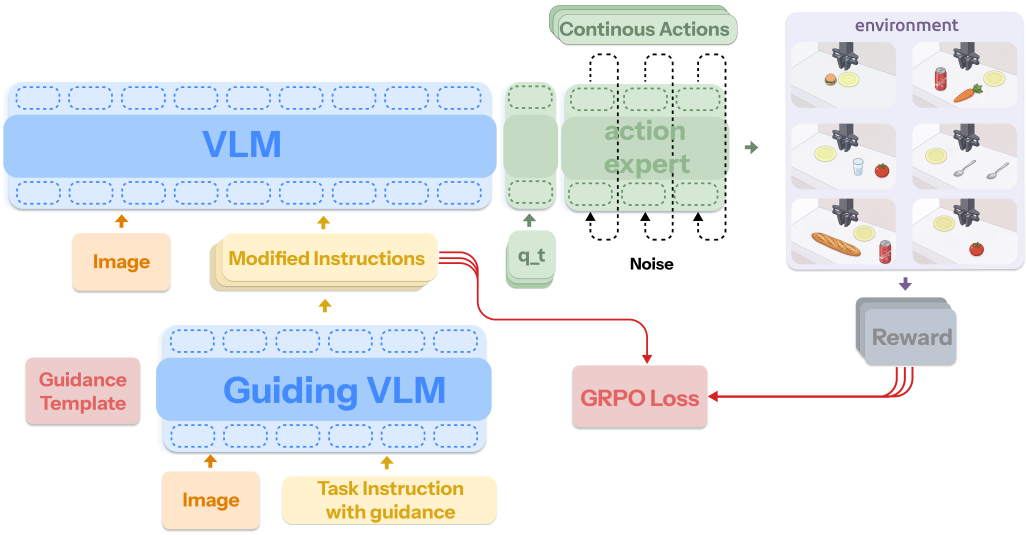

Figure 2: Model architecture and training loop. The proposed method inserts a scene-conditioned language-conditioning space policy before a frozen VLA action policy. Given the initial task instruction, the scene image, and a failure-derived command-space prior that specifies the desired command format, the language-conditioning space policy generates a VLA-grounded command using object appearance, spatial cues, and target-location information. The command is passed to the unchanged VLA policy for action prediction. During training, only the language-conditioning space policy is optimized: rollouts in the environment produce sparse task-completion rewards, which are used as the RL signal for improving command generation while keeping the VLA weights frozen.

behavior from a frozen one.

3.3 Problem Setting

We are given a pretrained frozen VLA policy π_{VLA} and seek to maximize return in $\mathcal{M}$ by selecting a command $\tilde{u} \in \mathcal{U}(u)$ rather than by changing action-model weights. Equivalently, the learned component chooses which member of the frozen family $\{\pi_{\text{VLA}}^{\tilde{u}} : \tilde{u} \in \mathcal{U}(u)\}$ will generate actions during the rollout. This formulation is useful only when the policy is sensitive to language-space variation, meaning that semantically valid commands can induce meaningfully different behavior. We assume only black-box access: for any observation o and command $\tilde{u}$, we may sample from $\pi_{\text{VLA}}(\cdot \mid o, \tilde{u})$, but we do not use weights, gradients, or internal activations.

4 Method

4.1 RL in Language-Conditioning space

In standard deployment, a VLA acts under the raw instruction u. We instead treat the command itself as the high-level decision variable: at observation o, we first select a command $\tilde{u} \in \mathcal{U}(u)$ and then sample the primitive action from $\pi_{\text{VLA}}(\cdot \mid o, \tilde{u})$. This induces an action-space transformation from primitive actions to language commands.

$$P^{\text{lang}}(s' \mid s, \tilde{u}) := \mathbb{E}_{\substack{o \sim Z(\cdot|s) \\ a \sim \pi_{\text{VLA}}^{\tilde{u}}(\cdot|o)}} \big[P(s' \mid s, a)\big],$$
$$r^{\text{lang}}(s, \tilde{u}) := \mathbb{E}_{\substack{o \sim Z(\cdot|s) \\ a \sim \pi_{\text{VLA}}^{\tilde{u}}(\cdot|o)}} \big[r(s, a)\big]. \tag{1}$$

These induced kernels define a transformed control problem, denoted $\mathcal{M}_u^{\text{lang}}$, whose action space is $\mathcal{U}(u)$ and whose dynamics and rewards are given by P^{lang} and r^{lang}. This is the sense in which the adaptation problem can be cast as reinforcement learning in language space. We focus on rollout-level optimization: a policy $q_\phi(\tilde{u} \mid o_0, u)$ is invoked once at the beginning of the episode, outputs a single command, and keeps it fixed for the full rollout. Sampling $\tilde{u} \sim q_\phi(\cdot \mid o_0, u)$ therefore selects one member of the frozen family $\{\pi_{\text{VLA}}^{\tilde{u}} : \tilde{u} \in \mathcal{U}(u)\}$.

$$J(\phi) = \mathbb{E}_{(o_0,u)\sim\mathcal{D}} \mathbb{E}_{\tilde{u}\sim q_\phi(\cdot|o_0,u)} \mathbb{E}_{\tau \sim p(\cdot|o_0, \pi_{\text{VLA}}^{\tilde{u}})} \big[R(\tau)\big]. \tag{2}$$

where $\mathcal{D}$ is the distribution of initial image-instruction pairs. The raw frozen-VLA baseline is recovered by the degenerate policy that always returns the original instruction.

4.2 Language Aliasing and Structured Exploration

Language-Conditioning space has additional structure that naive RL does not exploit. In particular, there may exist distinct commands $\tilde{u} \neq \tilde{u}'$ such that $\pi_{\text{VLA}}(\cdot \mid o, \tilde{u}) \approx \pi_{\text{VLA}}(\cdot \mid o, \tilde{u}')$ for many observations o; we refer to this as *language aliasing*. It is common for shallow rewrites, where grammatical edits or near-synonymous substitutions are lexically different but behaviorally redundant. Our method therefore biases exploration toward short executable commands that vary along grounding-relevant dimensions such as source-object description, target cues, color, shape, and spatial location; Figure 7 illustrates this distinction.

4.3 Optimizing the Guiding VLM Policy

We instantiate q_ϕ with a Guiding VLM and train it with GRPO (Shao et al., 2024) using only scalar rollout reward. Figure 2 summarizes the language-conditioning space optimization loop: the policy receives the initial image and instruction, proposes candidate commands, and the frozen VLA policy executes a rollout under each one. For each training pair (o_i, u_i), we sample a group of candidate commands $\{\tilde{u}_i^{(k)}\}_{k=1}^{G}$ from $q_\phi(\cdot \mid o_i, u_i)$, execute the frozen VLA under each command, and compare the resulting returns $\{R_i^{(k)}\}_{k=1}^{G}$ within the group. We use the normalized advantage

$$\hat{A}_i^{(k)} = \frac{R_i^{(k)} - \bar{R}_i}{\sigma_i + \varepsilon}, \qquad \bar{R}_i = \frac{1}{G}\sum_{k=1}^{G} R_i^{(k)}, \quad (3)$$

where σ_i is the empirical standard deviation of rewards in the group. Commands with larger relative advantage are up-weighted by the next update, while weaker ones are down-weighted. Appendix C summarizes the procedure.

4.4 VLA Black-Box Deployment

At inference time, the system receives a new image and instruction, samples one command $\hat{\tilde{u}}$ from the trained language-conditioning space policy, and conditions the frozen VLA policy on that command for the full rollout. There is no reward query, no RL update, and no action-model modification at test time. VLA Grounder changes only the language-conditioning input and requires only black-box forward access to the downstream action policy. In this paper, we evaluate this setup on frozen π_0 and OpenVLA backbones.

4.5 Why Language Exploration Matters

Group-based RL algorithms, such as GRPO, derive a signal by comparing multiple samples for the same task instance. If the sampled set differs only in grammar, then the reward cannot meaningfully separate better and worse conditioning signals. By contrast, if samples vary in robot-relevant grounding cues, for example, color, left-right position, object description, or target relation—the optimization process can assign credit to language transformations that actually change downstream behavior.

5 Experiments

We evaluate on two benchmark families. VL-Think (Kachaev et al., 2025) isolates visual-language reasoning under fixed manipulation: the robot places the same source object on a target board matching an abstract concept such as an arrow, sign, shape or weather icon. RL4VLA (Liu et al., 2025) tests broader semantic generalization; we use *MultiPlate*, *MultiCarrot*, and fixed-object *Pepper*, which require distractor handling and object-target grounding. All reported VLA backbones are frozen, and we report success rate as mean $\pm$ standard deviation.

The primary baseline is the frozen VLA conditioned on the raw human instruction. We also compare against the structured language-conditioning space policy before RL ("w/o GRPO"), generic command rewriting without structured reasoning ("no reasoning"), and structured-guidance variants with different language-conditioning space policy sizes. For Pepper, we include TextGrad (Yuksekgonul et al., 2024) and GEPA (Agrawal et al., 2025), two general-purpose prompt-optimization baselines that rewrite or mutate commands in language space. We organize the empirical section around main research questions:

RQ1: Does language-conditioning space optimization improve frozen VLA policies across backbones? Our experiments on VL-Think and RL4VLA (Table 1, Table 2, Figure 4) reveal a consistent effect: changing only the language input can substantially improve frozen action policies. The effect appears for both π_0 and OpenVLA, which suggests that the method is not exploiting a single backbone-specific quirk. The gains are most visible on VL-Think, where success depends heavily on converting abstract or symbolic targets into visually grounded commands. RL4VLA shows a smaller

Table 1: Main results on VL-Think. All VLA backbones are frozen. The language-conditioning space policy uses the strongest structured reasoning configuration, Qwen3.5-9B.

	π_0			OpenVLA			OpenVLA	
Task	orig	Qwen3.5-9B w/o GRPO	Qwen3.5-9B	orig	Qwen3.5-9B w/o GRPO	Qwen3.5-9B	TextGrad	GEPA
Arrow	4.2 ± 4.8	4.7 ± 0.0	$\mathbf{41.7 \pm 2.0}$	13.5 ± 3.2	21.4 ± 2.0	$\mathbf{62.5 \pm 1.3}$	44.0 ± 2.5	51.3 ± 2.8
Color	32.3 ± 6.4	25.0 ± 1.3	$\mathbf{40.1 \pm 4.5}$	55.7 ± 8.7	63.5 ± 4.5	$\mathbf{82.8 \pm 2.6}$	61.2 ± 3.6	73.2 ± 1.2
Laundry	12.5 ± 1.3	10.4 ± 0.7	$\mathbf{25.0 \pm 4.6}$	15.6 ± 2.6	21.9 ± 5.6	37.5 ± 3.8	19.1 ± 3.6	$\mathbf{47.0 \pm 2.1}$
Public Info	9.4 ± 1.3	14.6 ± 2.0	$\mathbf{43.8 \pm 2.2}$	21.4 ± 3.7	27.1 ± 3.2	$\mathbf{65.6 \pm 3.4}$	25.6 ± 1.6	48.5 ± 3.1
Shape	12.0 ± 5.9	21.4 ± 3.2	$\mathbf{43.2 \pm 7.8}$	27.6 ± 1.5	49.0 ± 7.7	$\mathbf{74.5 \pm 6.4}$	54.2 ± 3.3	59.5 ± 4.1
Traffic	5.7 ± 2.0	8.3 ± 2.7	$\mathbf{32.3 \pm 4.5}$	16.7 ± 2.7	28.1 ± 4.6	$\mathbf{63.0 \pm 4.8}$	35.8 ± 3.4	49.4 ± 5.1
Weather	12.0 ± 2.0	14.6 ± 2.7	$\mathbf{44.8 \pm 2.7}$	20.3 ± 4.6	31.3 ± 3.4	$\mathbf{61.5 \pm 3.9}$	30.4 ± 4.1	57.2 ± 3.4
Avg.	12.6	14.1	**38.7**	24.4	34.6	**63.9**	38.6	55.2

but still consistent trend, which is expected because these tasks add multi-object distractors and execution errors that cannot be removed by language alone.

RQ2: Does GRPO improve over the non-RL language-conditioning space policy and generic prompt-optimization baselines? The non-RL language-conditioning space policy already provides a useful command prior, but it does not reliably identify which command aliases the frozen action policy will execute. GRPO adds exactly this missing selection pressure: candidate commands are compared by downstream task reward rather than by linguistic plausibility. This distinction is clearest on VL-Think, where the trained policy separates sharply from its non-RL version. On RL4VLA the improvement is more modest, but remains positive despite higher action-side noise.

RQ3: How does reward-trained language optimization compare with traditional prompt optimization? We also tested generic prompt-optimization methods, including TextGrad and GEPA, using the same frozen-VLA setting. Our results in Table 1 show that these methods can

Figure 3: Trajectories comparison on RL4VLA MultiObject with frozen π_0. The default command underspecifies the object and causes unstable motion, while the rewritten command adds visible grounding cues and yields a more direct successful trajectory.

Figure 4: GRPO training for VL-Think tasks. Each panel reports rollout success rate over training steps for a symbolic grounding category. The upward trends indicate that reward updates to the language-conditioning space policy improve command selection while the downstream VLA action policy remains frozen.

improve over the raw instruction baseline, but not nearly as much as GRPO-based RL adaptation. The reason is qualitative as well as quantitative: generic prompt optimizers often modify punctuation, wording, or writing style in ways that are hard to interpret as better robot grounding. VLA Grounder instead updates the language-conditioning space policy from rollout reward, so it favors command changes that actually alter execution success rather than commands that only look cleaner as text.

RQ4: Do better commands lead to better VLA representations? We further probe whether improved commands merely change the final text input or also change the internal representations used by the VLA. Using VL-Think Public Info task, we compare OpenVLA representations under the original task instruction and under the rewritten command. The probing results Table 4, Figure 3 indicate that rewritten commands produce more

Table 2: Results on the RL4VLA semantic generalization tasks. The same structured language-conditioning space policy improves both frozen π_0 and frozen OpenVLA in multi-object and distractor-heavy settings.

	π_0			OpenVLA		
Task	orig	Qwen3.5-9B w/o GRPO	Qwen3.5-9B	orig	Qwen3.5-9B w/o GRPO	Qwen3.5-9B
MultiPlate	9.9 ± 1.5	10.4 ± 3.2	$\mathbf{17.2 \pm 2.2}$	54.7 ± 2.2	56.3 ± 4.6	$\mathbf{64.6 \pm 5.2}$
MultiCarrot	17.2 ± 5.1	21.4 ± 4.5	$\mathbf{29.7 \pm 5.6}$	59.4 ± 5.9	60.9 ± 2.6	$\mathbf{63.5 \pm 5.3}$
Avg.	13.6	15.9	**23.4**	57.0	58.6	**64.0**

Table 3: Common instruction-grounding failures and useful command rewrite dimensions. The learned policy improves frozen VLA behavior by replacing abstract target names, rare object categories, ambiguous references, or weak object names with visually grounded descriptions that expose color, shape, material, symbol, or spatial cues.

Failure type	Raw instruction	Useful rewrite dimension	Language policy rewrite
Abstract target	"sunrise icon"	visible symbol/color	"white card with yellow sun"
Rare object name	"champagne glass"	shape/material	"tall white glass"
Multi-object ambiguity	"plate"	spatial relation	"left yellow plate"
Weak object grounding	"bread"	color/shape	"brown round object"

Table 4: Linear probing accuracy on VL-Think Public Info tasks using OpenVLA hidden states under the default instruction and the rewritten command.

Prompt	Accuracy
Default	66.6
VLA Grounder	89.3

informative prompt-conditioned representations: task-relevant distinctions become easier to recover from the VLA hidden states, and this aligns with the downstream success improvements. This suggests that language-conditioning space optimization changes the conditioning representation seen by the action model, rather than only changing a superficial instruction string.

RQ5: What command transformations does the learned policy discover? Based on prompt evolution analysis during training (Table 3) we found that the learned policy tends to produce shorter and more executable commands rather than more verbose descriptions. On VL-Think, it often converts symbolic target labels into compact spatial or visual aliases, such as replacing an icon name with a reference to a white card at a particular position. On RL4VLA and Pepper, it usually keeps familiar receptacle names but rewrites brittle source-object names into visible color and shape descriptions. This pattern indicates that the policy is learning how to reduce the semantic burden placed on the frozen action model. The appendix provides Figure 5 prompt-format examples.

6 Discussion

6.1 What the Method Actually Adapts

The proposed method does not directly improve low-level motor competence. Instead, it changes how existing capabilities are accessed. This distinction matters both scientifically and practically: if a frozen VLA already contains useful action structure, then a substantial part of adaptation may be achievable by changing the VLA-grounded command that exposes that structure.

6.2 Command-Space Design and Model Scale

The ablations suggest that the command space and the language model play different roles. Generic rewriting can already help, which confirms that the raw instruction is often a weak conditioning signal. Structured guidance improves the search space by pushing candidates toward object identity, visual attributes, and spatial grounding rather than shallow paraphrases. Scaling the language model then becomes more useful because the model has a richer space of possible aliases to choose from. The benefit is not perfectly monotonic across every category, however, so scale alone is not the explanation; the strongest setting combines structured command variation with reward-based selection.

Task	π_0	+Qwen3-4B no reasoning	+Qwen3-4B reasoning	+Qwen3.5-4B reasoning	+Qwen3.5-9B reasoning
Arrow	4.2 ± 4.8	8.9 ± 1.0	16.2 ± 2.7	15.1 ± 2.6	$\mathbf{41.7 \pm 2.0}$
Color	$\mathbf{32.3 \pm 6.4}$	22.9 ± 3.0	25.0 ± 0.0	29.2 ± 8.7	40.1 ± 4.5
Laundry	12.5 ± 1.3	22.4 ± 1.0	20.8 ± 1.0	13.5 ± 2.0	$\mathbf{25.0 \pm 4.6}$
Public Info	9.4 ± 1.3	19.8 ± 1.5	23.4 ± 1.3	24.0 ± 1.5	$\mathbf{43.8 \pm 2.2}$
Shape	12.0 ± 5.9	21.9 ± 3.4	27.1 ± 3.7	29.2 ± 3.0	$\mathbf{43.2 \pm 7.8}$
Traffic	5.7 ± 2.0	21.9 ± 1.3	21.9 ± 2.6	13.0 ± 2.0	$\mathbf{32.3 \pm 4.5}$
Weather	12.0 ± 2.0	25.0 ± 1.3	22.4 ± 3.2	21.9 ± 2.2	$\mathbf{44.8 \pm 2.7}$
Avg.	11.0	21.1	23.9	23.3	**38.8**

Table 5: Ablation on frozen π_0 over VL-Think benchmark. "No reasoning" denotes a generic command-rewriting policy; "reasoning" denotes the structured guidance used by our method.

6.3 Relation to Other Post-Training Strategies

Language-Conditioning space optimization is complementary to direct VLA fine-tuning, reward-model training, and verifier-based test-time selection. Fine-tuning can change the action policy itself; verifier methods can select better candidates at inference time; our approach improves the upstream command that conditions the frozen action policy. In this sense, language-conditioning space optimization can be viewed as a lightweight post-training primitive that may compose naturally with broader VLA post-training pipelines.

6.4 Limits of Language-Conditioning Space Optimization

The gains are largest when the user instruction is semantically valid but visually under-specified for the action policy. Symbolic VL-Think tasks such as arrows, public-information signs, weather icons, shapes, and traffic signs benefit because the learned policy can replace the symbolic target with a simpler visual or spatial description. Direct perceptual categories such as Color leave less room for improvement because the raw instruction already exposes a usable cue. The method is also limited on tasks where language is not the only bottleneck: clutter, distractors, wrong object grounding, severe visual misperception, missing low-level manipulation skill, or long-horizon reasoning demands can still dominate the rollout outcome.

6.5 Broader Implication

The broader implication of this work is that language can function as an optimizable adaptation layer for robot foundation models. If this view is correct, then future post-training of VLAs need not operate only in weight space or action space; it can also operate in the space of semantic conditioning signals that mediate between human intent and embodied action.

7 Conclusion

We study adaptation of frozen VLA policies through the language-conditioning input rather than through action-weight updates. The resulting perspective treats language as an optimizable conditioning variable and casts improvement as reinforcement learning in language space. VLA Grounder combines a language-conditioning space policy, a failure-derived command-space prior, and a sparse downstream task reward to discover VLA-grounded commands that better expose latent capability in frozen action policies. The central takeaway is simple but consequential: for large robot foundation models, policy improvement can be lifted from action space into language space.

Limitations

Our results show that language-conditioning space optimization can improve frozen VLA policies, but the scope of the evidence is still limited. The method also cannot repair all failures of a frozen action policy. Language-space optimization should be viewed as an adaptation layer over existing VLA capabilities rather than as a replacement for improving perception, robustness, control, or long-horizon reasoning.

Ethical Considerations

This work studies language-conditioned robot control in benchmark settings and does not involve human-subject data.

Artifact Licenses

We use publicly available benchmark and model artifacts, including VL-Think, RL4VLA, OpenVLA, π_0, and Qwen, under their respective licenses and terms of use. Any released code or checkpoints will be distributed with an explicit license. Our use of these artifacts is limited to research evaluation.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, and 1 others. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*.

Kevin Black and 1 others. 2024. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, and 35 others. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *Preprint*, arXiv:2307.15818.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. 2026. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models. *Preprint*, arXiv:2510.25889.

Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, and 275 others. 2025. Open x-embodiment: Robotic learning datasets and rt-x models. *Preprint*, arXiv:2310.08864.

Bear Häon, Kaylene Stocking, Ian Chuang, and Claire Tomlin. 2025. Mechanistic interpretability for steering vision-language-action models. *Preprint*, arXiv:2509.00328.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, and 37 others. 2025a. $\pi^*_{0.6}$: a vla that learns from experience. *Preprint*, arXiv:2511.14759.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, and 17 others. 2025b. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *Preprint*, arXiv:2504.16054.

Nikita Kachaev, Mikhail Kolosov, Daniil Zelezetsky, Alexey K Kovalev, and Aleksandr I Panov. 2025. Don't blind your vla: Aligning visual representations for ood generalization. *arXiv preprint arXiv:2510.25616*.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024. Openvla: An open-source vision-language-action model. *Preprint*, arXiv:2406.09246.

W. Kong and 1 others. 2024. Prewrite: Prompt rewriting with reinforcement learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 594–601.

M. Kwon and 1 others. 2024. Stableprompt: Automatic prompt tuning using reinforcement learning for large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

D. Lee, S. Song, and 1 others. 2023. Read-only prompt optimization for vision-language few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

J. Liu and 1 others. 2025. What can rl bring to vla generalization? an empirical study. *arXiv preprint arXiv:2505.19789*.

Daria Pugacheva, Andrey Moskalenko, Denis Shepelev, Andrey Kuznetsov, Vlad Shakhuro, and Elena Tutubalina. 2025. Bring the apple, not the sofa: Impact of irrelevant context in embodied ai commands on vla models. *Preprint*, arXiv:2510.07067.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. 2024. Bridgedata v2: A dataset for robot learning at scale. *Preprint*, arXiv:2308.12952.

Selma Wanna, Agnes Luhtaru, Jonathan Salfity, Ryan Barron, Juston Moore, Cynthia Matuszek, and Mitch Pryor. 2026. Limited linguistic diversity in embodied ai datasets. *Preprint*, arXiv:2601.03136.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. 2024. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. 2025. Robotic control via embodied chain-of-thought reasoning. *Preprint*, arXiv:2407.08693.

Zhihao Zhan, Yuhao Chen, Jiaying Zhou, Qinhan Lyu, Hao Liu, Keze Wang, Liang Lin, and Guangrun Wang. 2026. Stable language guidance for vision-language-action models. *Preprint*, arXiv:2601.04052.

Y. Zhang, Y. Qi, and X. Zheng. 2025. Experiences from benchmarking vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2511.11298*.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. 2023. Learning fine-grained bimanual manipulation with low-cost hardware. *Preprint*, arXiv:2304.13705.

Y. Zhong and 1 others. 2025. A survey on vision-language-action models: An action tokenization perspective. *arXiv preprint arXiv:2507.01925*.

Y. Zhou and 1 others. 2024. Archer: Training language model agents via hierarchical multi-turn rl. In *International Conference on Machine Learning (ICML)*.

Viktoriia Zinkovich, Anton Antonov, Andrei Spiridonov, Denis Shepelev, Andrey Moskalenko, Daria Pugacheva, Elena Tutubalina, Andrey Kuznetsov, and Vlad Shakhuro. 2025. Sparta: Evaluating reasoning segmentation robustness through black-box adversarial paraphrasing in text autoencoder latent space. *Preprint*, arXiv:2510.24446.

A Appendix Overview

This appendix collects the implementation and qualitative material that supports the main paper. Appendix B shows the prompt format used by the language-conditioning space policy and clarifies the output contract: the model may reason about the scene internally, but only the final VLA-grounded command is passed to the frozen action policy. Appendix **??** provides a trajectory-level example comparing the original instruction with an optimized command. Appendix C summarizes the GRPO training procedure and the black-box optimization loop, and Appendix D reports the recovered model, adapter, and optimization parameters used in the archived runs.

B Prompt Format Example

Figure 5 illustrates the prompt format used to transform a human instruction into a VLA-grounded command. The input consists of the current scene image, the original task instruction, and a structured instruction that asks the language model to identify the source object, target, and visible grounding cues. The policy is allowed to use this reasoning to choose the command, but the downstream VLA receives only the final text inside the `<answer>` field. This separation prevents long explanations from entering the action model while preserving the benefit of scene-aware language reasoning.

C Training Algorithm and Optimization Loop

The training procedure treats command generation as the only learned component. For each image-instruction pair, the language-conditioning space policy samples a group of candidate commands. Each candidate is executed by the same frozen VLA policy, and the resulting sparse rollout rewards are compared within the group. GRPO then updates only the language policy, leaving the action model unchanged.

Algorithm 1: VLA Grounder Training
Input: training distribution $\mathcal{D}$ over image-instruction pairs (o, u), frozen VLA policy π_{VLA}, language-conditioning space policy q_ϕ, GRPO group size G.
For each training instance:

1. Sample a GRPO group of commands $\{\tilde{u}^{(k)}\}_{k=1}^{G}$ from $q_\phi(\cdot \mid o, u)$.
2. For each $\tilde{u}^{(k)}$, run a rollout with the same frozen VLA policy, keep the command fixed for the rollout, and collect scalar return $R^{(k)}$.
3. Compute normalized within-group advantages from $\{R^{(k)}\}_{k=1}^{G}$.
4. Apply a GRPO update to q_ϕ using the grouped commands and advantages; keep π_{VLA} unchanged.

Output: trained language-conditioning space policy q_ϕ.

D Training Parameters

This section reports the implementation details recovered from the archived training artifacts (Table 7, Table 6) . The interface model is adapted with LoRA while the downstream VLA policy is kept frozen. The two archived VL-Think runs share the same model and optimization configuration, differing only in the downstream frozen backbone and output directory.

Field	Recovered value
Base language model	`Qwen/Qwen3.5-9B`
PEFT method	LoRA
LoRA rank r	32
LoRA scaling α	64
LoRA dropout	0.05
Train-time precision flag	`bf16 = true`
Target modules	12 projection modules spanning attention and MLP blocks
Tokenizer context length	`model_max_length = 262144`
Tokenizer padding / truncation	left padding, left truncation

Table 6: Language-conditioning space policy configuration

Field	π_0/VL-Think archive	OpenVLA/VL-Think archive
Gradient accumulation steps	8	8
Max training steps	1600	1600
Learning rate	5×10^{-6}	5×10^{-6}
Scheduler	cosine	cosine
Stored `warmup_steps` field	0.05	0.05
Weight decay	0.01	0.01
Adam β_1, β_2	0.9, 0.999	0.9, 0.999
Adam ϵ	10^{-8}	10^{-8}
Max grad norm	1.0	1.0
Per-device eval batch size	8	8
Save strategy / save steps	steps / 500	steps / 500

Table 7: GRPO optimization configuration.

Full Prompt and Example Output

Exact prompt prior, runtime scene-conditioned input, and concrete model response. The language policy may reason in <think>, but only <answer> is sent to the frozen VLA policy.

fixed prior | image + original task | <think> may reason | <answer> goes to VLA

role: system

fixed across tasks

You are a language translation assistant for robot manipulation tasks. You translate human instructions into commands a Vision-Language-Action (VLA) robot can act on.

You will be given:
- A user-provided instruction describing a manipulation goal and an image of the scene.

Your task is to:
1. Understand the meaning of the original instruction.
2. Correctly identify the object to pick and the target location to place it, then reason about how the robot visually perceives each — these may not be the same as your own understanding.
3. Generate a rephrased command that helps the robot locate and act on the target.

Keep in mind: the VLA perceives the scene visually and may not recognize objects the same way you do. Your translation must bridge that perceptual gap.

Feel free to explore.

role: user

current image and original instruction

[image: current scene]

Task: {txt}.
Generate a command for the robot to successfully complete the task.
When describing the object or the target, always reason from what the robot likely sees in the image, not from your own understanding of the object. The rules below are all applications of this.

Rules for the command:
1. Always keep the object name (e.g. 'carrot'), unless the object is unrecognizable to the robot — in that case replace the name with a visual description per Rule 7.
2. One short sentence. No line breaks, no numbering.
3. Try different grounding cues to help locate the object or the target — for example: spatial words (left, right, center, far, near), visual properties (color, size, shape), or relative position (closest, next to the cup). Try combining cues (e.g. 'far right', 'small red one on the left'). Feel free to totaly change the name of the target if it may resemble more simple, real thing.
4. Use only simple, everyday words.
5. If the object to pick or the placement target is something the robot is unlikely to recognize by name (abstract concept, brand, celebrity, symbol), describe it purely by its visual appearance in the scene — what the robot would actually see it as.

Think inside <think></think> tags following these steps:
1. Identify the object to pick and the placement target in the scene.
2. Reason about how the robot likely perceives each.
3. Decide how to rephrase the command.
Keep thinking process short, no more than 10 sentences.
Then give your final command inside <answer> your_command_here </answer> tags.
NOTE: only the command inside <answer></answer> will be passed to VLA!

model output example

reasoning plus final command

MODEL OUTPUT: REASONING
<think>
The user wants me to translate the instruction "put carrot on sunrise icon" into a command for a robot.
1. Identify the object to pick: There is an orange object with a green top on the table. It looks like a carrot. The robot might see it as an orange carrot-shaped object or just carrot. Given Rule 1, I should keep carrot if recognizable, which it likely is.
2. Identify the placement target: There are three white cards on the table. Left card: grey squiggly lines. Center card: yellow semi-circle with rays behind horizontal lines; this looks like a sun setting or rising. Right card: cloud with blue rain drops. The target is the center card with the sun symbol. The robot sees a white square card with a yellow sun symbol.
3. Reasoning about perception: The robot sees a carrot lying on the table. The robot sees three cards. The one in the middle has a yellow sun drawing. The command needs to direct the robot to move the carrot to that specific card.
4. Drafting the command: "Put the carrot on the sun card" is a bit vague. "Pick up the orange carrot and place it on the white square with the yellow sun" is more descriptive. "Place the carrot on the center card with the sun picture" adds a spatial cue. "Put the carrot on the card with the yellow sun and black rays" is very specific.
The rule asks for grounding cues: spatial words and visual properties. "Sunrise icon" might be abstract for a robot, while "white card with the yellow sun" is concrete. The object name carrot should remain because it is recognizable. Final choice should be one short sentence with simple words.
</think>

<answer>
Place the carrot on the white card with the yellow sun symbol.
</answer>

Figure 5: Prompt-format example for VLA Grounder. The language policy receives the scene image and original task, identifies visual cues for the source object and target, and emits one concise VLA-grounded command. Only the command inside the <answer> tags is passed to the frozen VLA policy; intermediate reasoning is used only to select a better command.

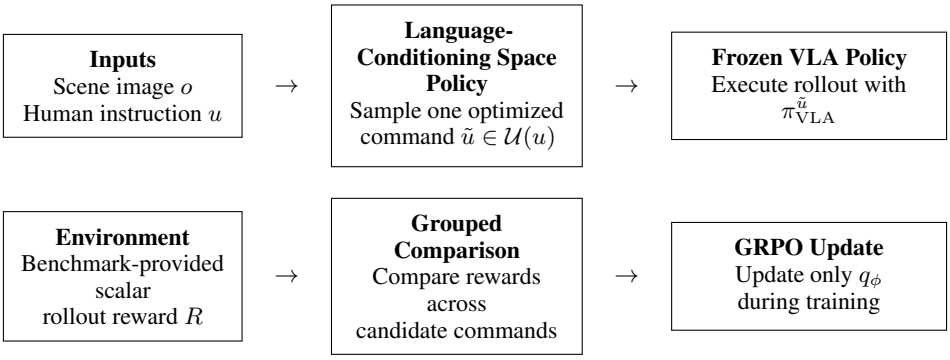

Figure 6: VLA Grounder language-conditioning space optimization loop. The image and user instruction define a context in which the language-conditioning space policy selects a command. The frozen VLA policy then executes a rollout under the command-conditioned action model $\pi_{\mathrm{VLA}}^{\tilde{u}}$. During training, grouped reward comparisons update only the language-conditioning space policy; during inference, the same composition is used without reward queries or parameter updates.

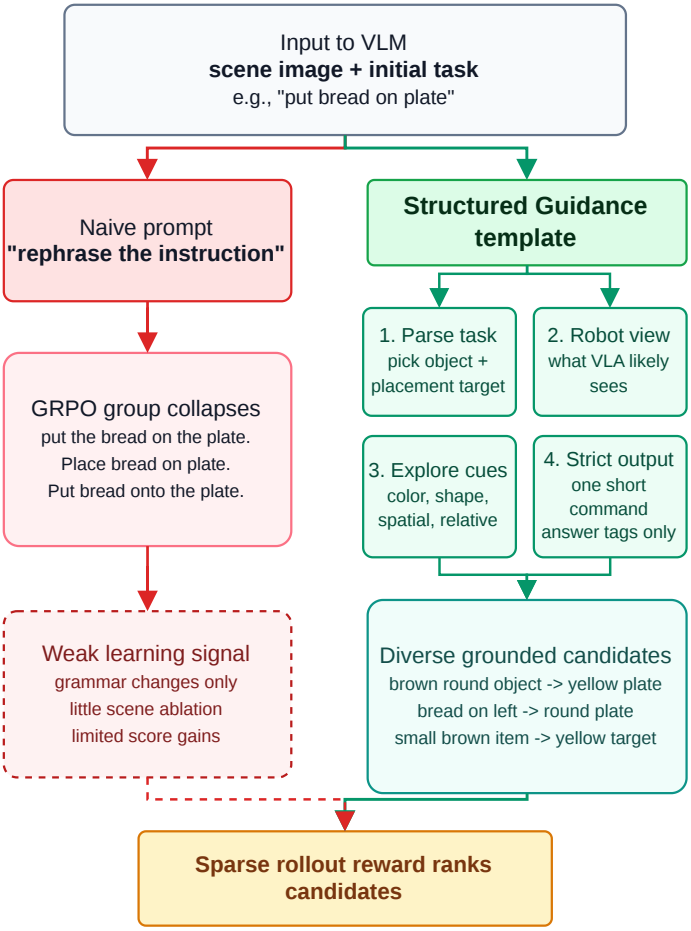

Figure 7: Language aliasing in command space. Naive rewriting often produces commands that differ only in surface form, so reward comparisons are weak because the frozen action policy behaves similarly under each candidate. VLA Grounder instead biases exploration toward short executable commands that vary along behaviorally meaningful axes such as source-object description, target cues, visible attributes, and spatial relations.